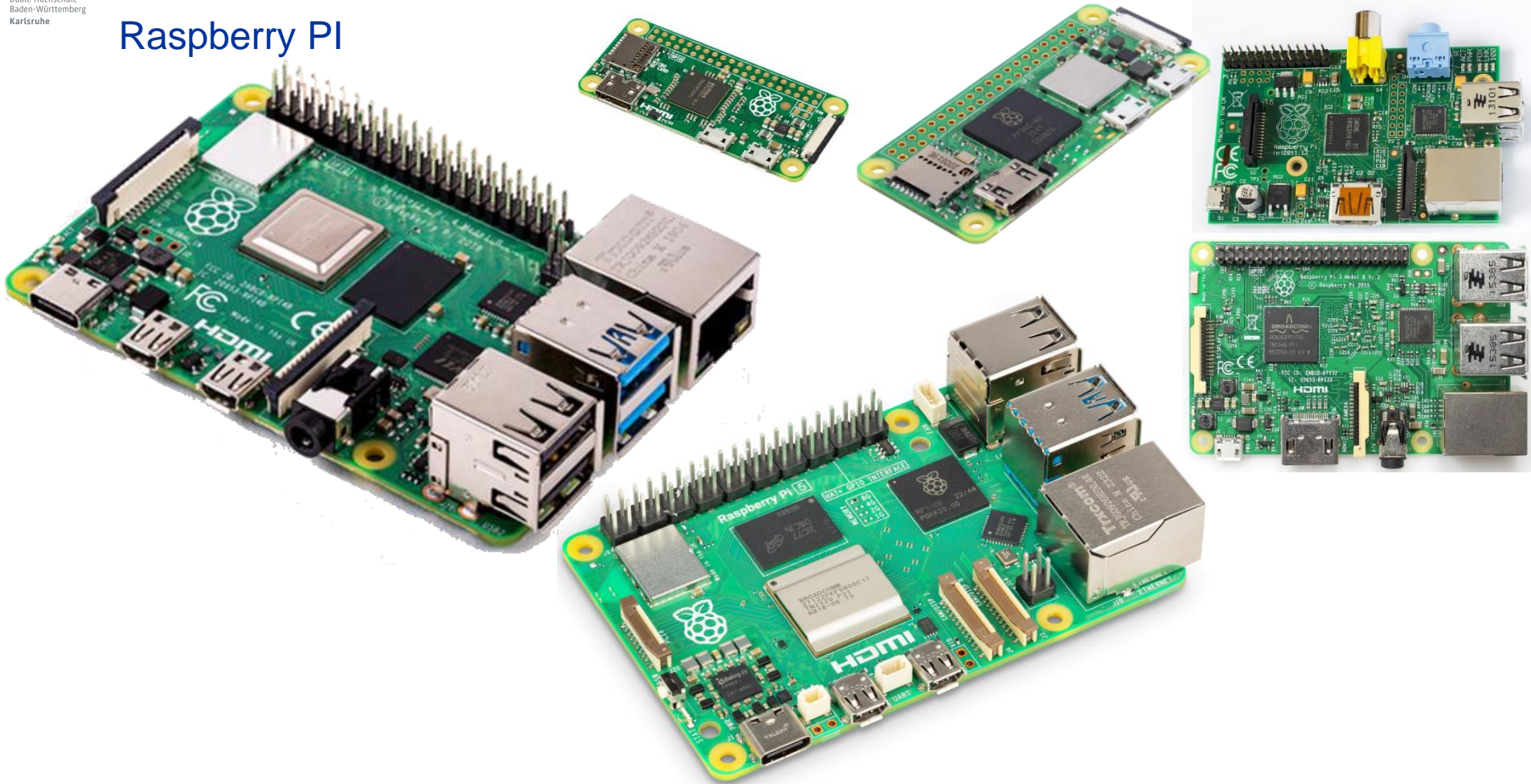




# Rechnersysteme

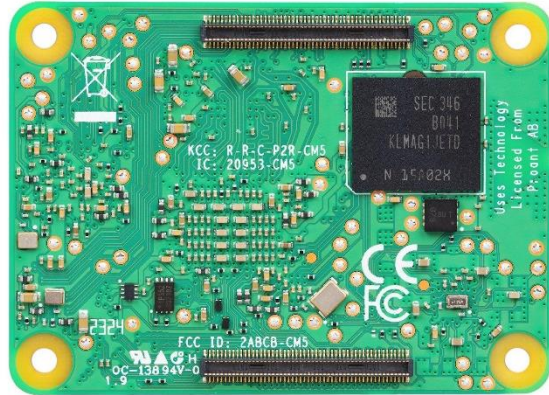
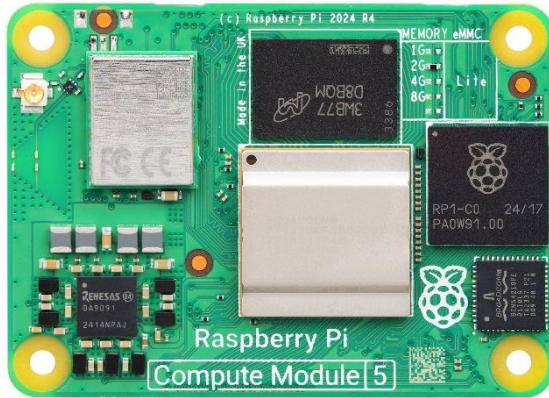
Dozent Dipl.-Ing. Dirk Hotz

# Raspberry Pi

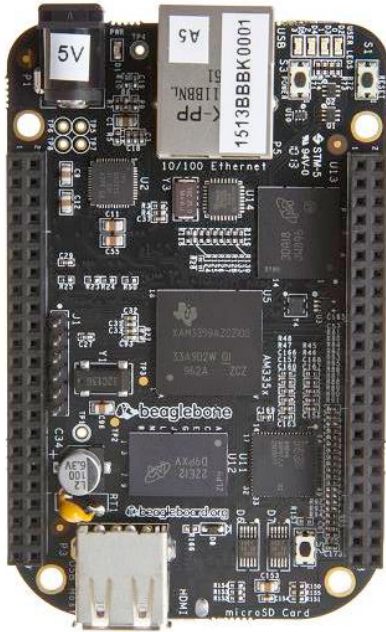




# Raspberry Pi



## Weitere Einplatinencomputer



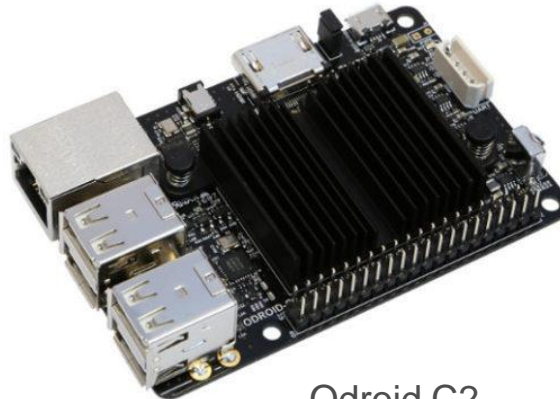
BeagleBone Black



Banana Pi



Banana Pi BPI-M6



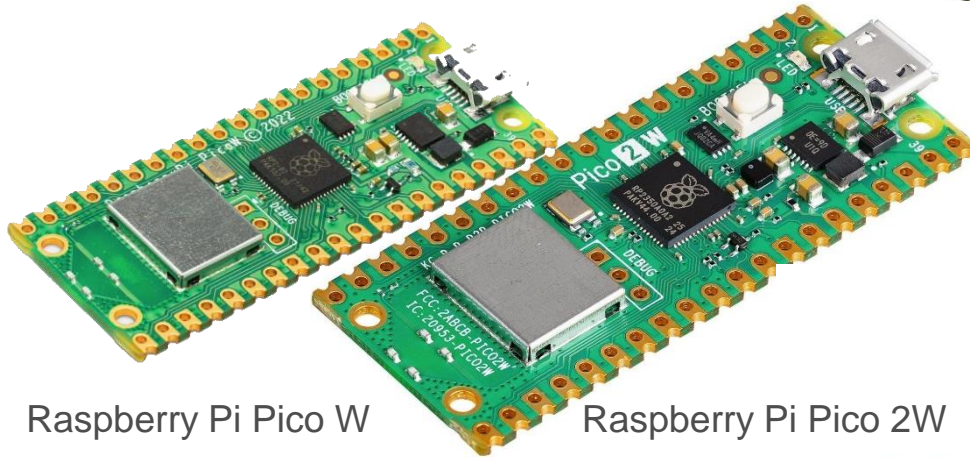
Odroid C2



Odroid M1

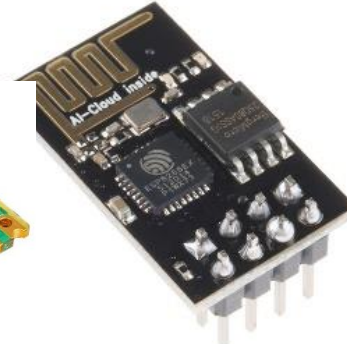


# Mikrocontrollerboards



Raspberry Pi Pico W

Raspberry Pi Pico 2W



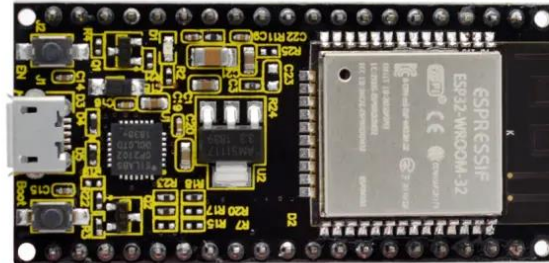
ESP 8266



Arduino Uno R3

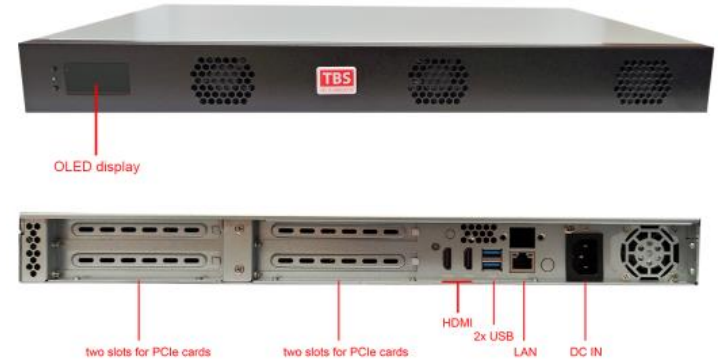
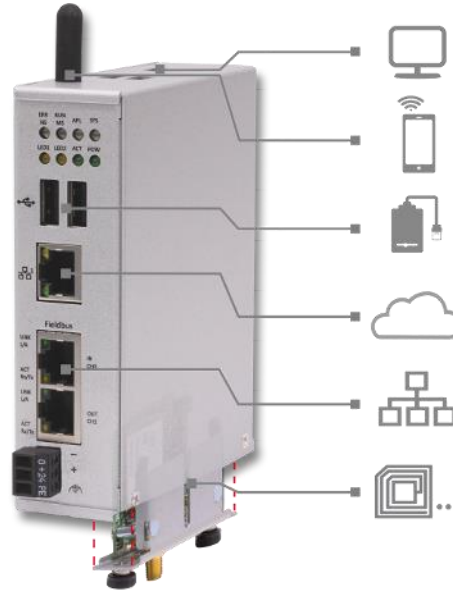


BBC micro:bit



ESP32

# Raspberry im Automatisierungsumfeld



# ARM Übersicht

## Architektur ARM-Design(s) / Familie(n)

ARMv1	ARM1	1985
ARMv2	ARM2, ARM3	1986, 1989
ARMv3	ARM6, ARM7	1991, 1993
ARMv4	ARM7TDMI, ARM8, StrongARM ARM9TDMI	1995, 1997
ARMv5	ARM7EJ, ARM9E, ARM10E	2002

# ARM Übersicht

Application  
**Microcontroller**  
Realtime

Architektur	ARM-Design(s) / Familie(n)	
ARMv6	ARM11 (1176, 11 MPCore, 1136, 1156) ARM Cortex-M (M0, M0+, M1)	Raspberry 1 / Zero BBC Micro:bit, Raspberry Pico (M0+)
ARMv7	ARM Cortex-A (A8, A9, A5, A15, A7, A12, A17) ARM Cortex-M (M3, M4, M7) ARM Cortex-R (R4, R5, R7, R8)	Raspberry 2 (A7), BeagleBone (A8) MCBSTM32F200 (M3)
ARMv8	ARM Cortex-A (A32, A53, A57, A72, A35, A73, A55, A75, A76, A77, A78, X1) ARM Cortex-M (M23, M33) ARM Cortex-R (R52) Arm Neoverse (E1, N1, V1)	Raspberry 3 (A53), Raspberry 4 (A72), Raspberry 5 (A76) Raspberry Pi Pico 2 (M33)
ARMv9	Arm Cortex-A (A510, A710, X2) Arm Neoverse (N2)	



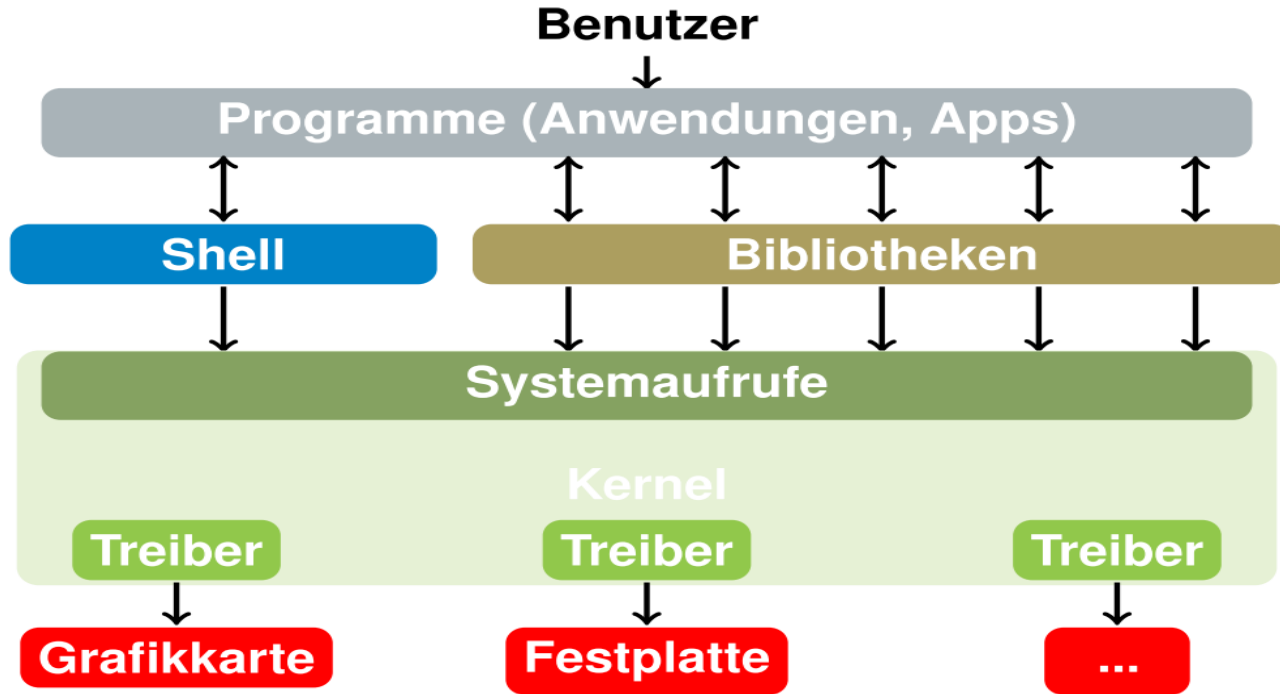
## Raspberry PI 4

- Broadcom BCM2711, Cortex-A72
- 64-bit SoC @ 1.5 GHz
- 2 GB LPDDR4 SDRAM (1 GB, 4 GB, 8 GB)
- 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac wireless
- LAN, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports
- Extended 40-pin GPIO header
- 2 × micro-HDMI ports (4k)
- MIPI DSI display port
- MIPI CSI camera port
- 4 pole stereo output and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode), OpenGL ES 3.1, Vulkan 1.0
- Micro SD
- 5 V/3 A DC via USB-c connector
- 5 V DC via GPIO header
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
- Operating temperature, 0–50°C

## Raspberry PI 5

- Broadcom BCM2712 quad-core 64-bit Arm Cortex-A76 CPU
- 64-bit SoC @ 2.4 GHz
- LPDDR4X-4267 SDRAM (2 GB, 4 GB, 8 GB, 16 GB)
- 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac wireless
- LAN, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports
- Extended 40-pin GPIO header
- 2 × micro-HDMI ports (4kp60)
- 2 × 4-lane MIPI camera/display transceivers
- PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter)
- H.265 (4kp60 decode), OpenGL ES 3.1, Vulkan 1.2
- Micro SD
- 5 V/5 A DC via USB-C connector
- Real-time clock (RTC), powered from external battery
- Power button
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
- Operating temperature, 0–50°C

## Betriebssystem - OS



- Vermittler zwischen Benutzer, Programmen und Hardware
- Kontrolle von Zugriffsrechten

## Woher bekomme ich Linux?

- Es gibt nicht ein, sondern viele Linuxe — Distributionen
- Live-CDs: Ausprobieren ohne Installation
- Dual-Boot: Installation parallel mit anderen OS
- Achtung: Manchmal lässt Windows nicht genügend Festplattenspeicher frei — dann löschen die Installer es nach einer Nachfrage!

## Distributionen

- (K)Ubuntu - benutzerfreundlich, mit Gnome/Unity bzw. KDE-GUI  
<http://www.ubuntu.com>, <http://www.kubuntu.org>
- OpenSuSE: sehr benutzerfreundlich, einfache Administration  
<http://www.opensuse.org>
- Redhat Enterprise Linux ( plus Klone: Scientific Linux, CentOS, OEL, AlmaLinux, Rocklinux)
- Debian
- ArchLinux



# Distribution für den Raspberry Pi



## Raspberry Pi OS (32-bit)

A port of Debian Bookworm with the Raspberry Pi Desktop

Veröffentlicht: 2024-11-19

Online - 1.1 GB Download



## LibreELEC

A Kodi Entertainment Center distribution



## OSMC

A fast and feature filled open source media center



## Volumio

The Audiophile Music Player and Streamer



## moOde audio player

Audiophile music streamer for the Raspberry Pi



## Ubuntu Desktop 24.10 (64-bit)

Desktop OS for RPi 4/400/5 models with 4Gb+

Veröffentlicht: 2024-10-10

Online - 2.7 GB Download



## Homebridge

Turn your Raspberry Pi into a HomeKit smart home bridge.



## Home Assistant

Open source home automation that puts local control and privacy first.



## Raspberrymatic

Lightweight Linux OS for running a HomeMatic/homematicIP IoT central.



## nymea

Smart Home/IoT platform, easy to use, open-source and privacy-focused.



## Gladys Assistant

A privacy-first, open-source home assistant that runs on the Raspberry Pi.



## openHAB

A vendor and technology agnostic open source automation software for your home.



## RetroPie

Turn your Raspberry Pi into a retro-gaming machine



## Recalbox

The retro-gaming OS supporting 100+ gaming systems!



## Lakka

The DIY retro emulation console



## Anthias

The most popular open source digital signage project



## Kali Linux

Kali Linux is an open-source, Debian-based Linux distribution geared towards various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering.



## FullPageOS

Display a full page browser on boot in kiosk mode



## MoodleBox

MoodleBox is an open source distribution combining a wireless access point with a full featured Moodle server.



## OctoPi

A Raspberry Pi distribution for 3D printers. Ships OctoPrint out-of-the-box.



## OctoKlipperPi

Includes the OctoPrint host software for 3d printers and Klipper 3D printer firmware service



## Mainsail OS

Klipper Firmware & Moonraker API and Mainsail UI - ready to print!



## SimplyPrint - 3D print anywhere, smarter

Effortlessly manage 1, 2 or 100 3D printers from anywhere with the SimplyPrint ecosystem - free and easy to set up (SimplyPrint plugin on top of OctoPrint)

## Aufgabe 1

- Raspberry anschließen und OS aufspielen
- Betriebssystem: [Raspberry Pi OS](https://www.raspberrypi.com/software/)  
<https://www.raspberrypi.com/software/>
- Raspberry PI Imager, Etcher, Rufus

## Netzwerk Konfigurieren

- /etc/network/interfaces

auto eth0

iface eth0 inet static

address 10.44.7.[101-116]

netmask 255.128.0.0

gateway 10.0.0.1

- /etc/resolv.conf

nameserver 10.3.15.32

nameserver 10.3.43.32

## Warum stimmt die Uhrzeit nicht?

- Raspberry hat keine RTC (Real Time Clock) eingebaut  
(ab Version 5 vorhanden, falls man noch zusätzlich eine Batterie/Akku als Pufferspeicher anschließt)

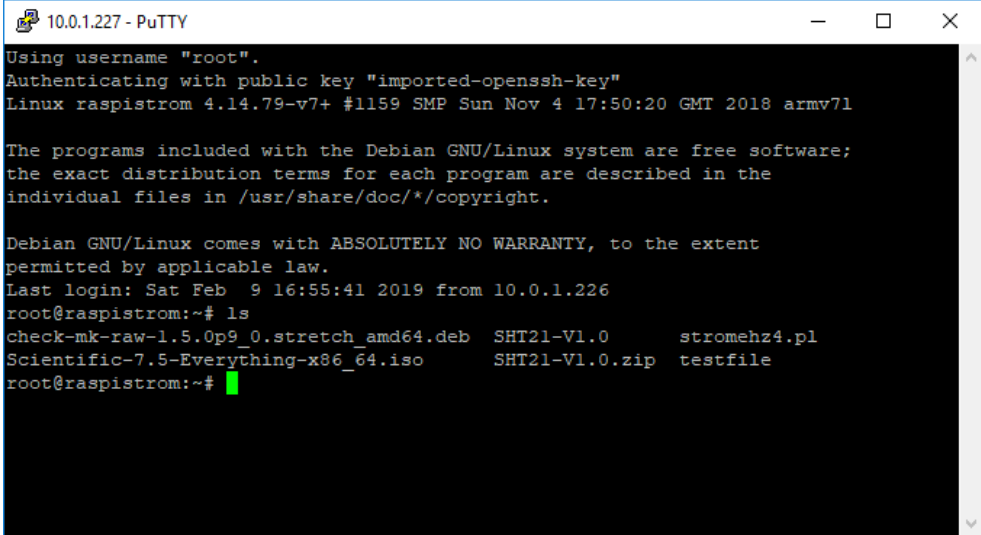
Lösung ist die aktuelle Uhrzeit beim Start übers Netzwerk zu beziehen, dafür verwendet man in der Regel NTP (Network Time Protocol)

Die DHBW blockiert den Port ins Internet (UDP 123), betreibt aber einen eigenen Zeitserver 10.0.0.1

```
sudo nano /etc/systemd/timesyncd.conf  
NTP=10.0.0.1
```



# Terminal und Shell



A screenshot of a PuTTY terminal window titled "10.0.1.227 - PuTTY". The terminal shows a login sequence for the user "root" on a Linux raspistrom system. The output includes the system version, a disclaimer about Debian GNU/Linux software, the last login time, and the result of a 'ls' command listing files in the home directory. A green cursor is visible at the end of the last prompt.

```
10.0.1.227 - PuTTY
Using username "root".
Authenticating with public key "imported-openssh-key"
Linux raspistrom 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Feb  9 16:55:41 2019 from 10.0.1.226
root@raspistrom:~# ls
check-mk-raw-1.5.0p9_0.stretch_amd64.deb  SHT21-V1.0      stromehz4.pl
Scientific-7.5-Everything-x86_64.iso      SHT21-V1.0.zip  testfile
root@raspistrom:~#
```

- Shell in einem Terminal
- Terminal: Tastatur-Eingabe und Zeichen-Ausgabe
- Shell: Startet und verwaltet Programme
- Philosophie: Eine Aufgabe — ein Programm
- Komplexität durch Verknüpfen von Programmen
- Funktioniert genauso auch per Netzwerk

## Grundlegende Shell-Benutzung

- Cursor-Up/Down: Vorherige Befehle wiederholen
  - Tabulator: Automatische Ergänzung von Dateinamen
  - Zeilen sind editierbar
  - Linke Maustaste: Markieren - Mittlere: Einfügen (putty= rechte Maustaste)
  - Control-a/e: Anfang/Ende der Zeile
  - Genaueres hängt vom Shell-Typ ab (sh, bash, csh,...)
- 
- Virtuelle Terminal: screen, tmux

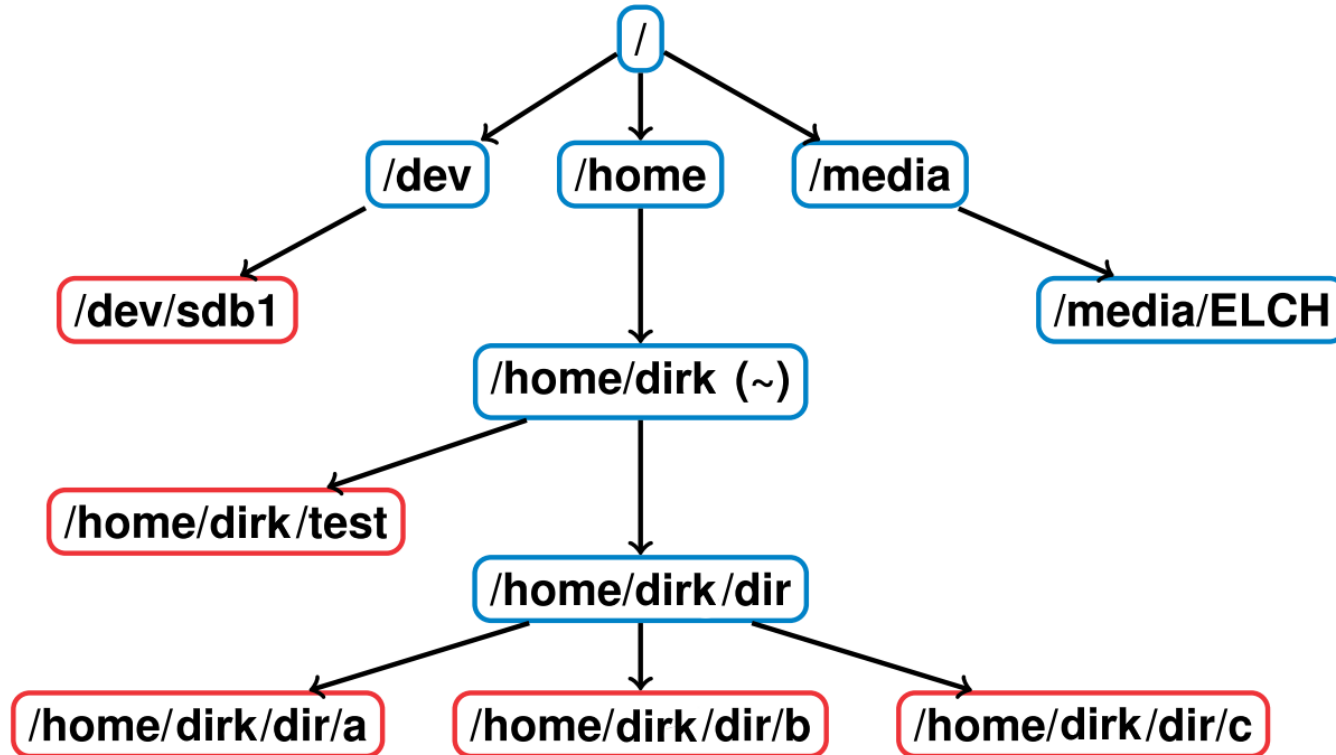
# Dateien

- Alle Daten werden in Dateien gespeichert
- Dateien können in Verzeichnissen zusammengefasst werden
- Eine Datei wird durch Ihren Pfad identifiziert
- Besondere Pfade:

.	aktuelles Verzeichnis
..	übergeordnetes Elternverzeichnis
~	eigenes Benutzerverzeichnis (Home)
~name	Benutzerverzeichnis (Home) des Benutzersname

- Dateien, die mit „.“ beginnen, sind versteckt
- Die Shell (und jedes andere Programm) hat ein aktuelles Arbeitsverzeichnis
- Pfade, die nicht mit „ / “ oder „ ~ “ anfangen, sind relativ zum Arbeitsverzeichnis

# Dateibaum





## Beispiele: Pfade

- `~/.local/share/Trash`  
Ort des Mülleimers im eigenen Home-Directory, das `.local`-Verzeichnis ist versteckt
- `/home/dirk/.local/share/Trash`  
Dasselbe Verzeichnis, wenn das Home-Directory `/home/dirk` ist
- `.local/share/Trash`  
Auch dasselbe Verzeichnis, wenn man im Home-Directory ist
- `share/Trash`  
Auch dasselbe Verzeichnis wenn man im Verzeichnis „`~/.local`“ ist

## Grundlegende Dateisystembefehle

<code>man &lt;program&gt;</code>	Hilfe, verlassen mit q
<code>ls &lt;file&gt;...</code>	Datei(en) auflisten
<code>cp [-r] &lt;src&gt;... &lt;dst&gt;</code>	Dateien kopieren (-r: rekursiv)
<code>mv &lt;src&gt;... &lt;dst&gt;</code>	Dateien verschieben/umbenennen
<code>rm [-r] &lt;file&gt;...</code>	Dateien löschen
<code>pwd</code>	aktuelles Verzeichnis ausgeben
<code>mkdir &lt;dir&gt;...</code>	Verzeichnis erzeugen
<code>cd &lt;dir&gt;...</code>	das Arbeitsverzeichnis wechseln

# Textdateien

- unter LINUX wird fast alles über Textdateien gesteuert
- Textviewer:
  - cat, less
- Texteditor Terminal:
  - vi(m), nano, mcedit, emacs
- Texteditor graphisch:
  - kate, kwrite, gedit, (x)emacs, nedit, eclipse, idle, atom, ...

# Was ist wo? – Verzeichnisstruktur 1

/	Wurzel (root-) Verzeichnis
/bin	Binaries, grundlegende Programme (ls, cat, cp, mount, ...)
/boot	Boot Loader inkl. Kernel
/dev	Gerätedateien (/dev/sda1, /dev/null, /dev/tty)
/etc	Systemkonfigurationsfiles
/home	Benutzerverzeichnisse
/lib	Bibliotheken
/media	Temporär eingehängte Medien (CD, USB-Massenspeicher, ...)
/mnt	Temporär eingehängte Dateisysteme
/opt	Zusätzliche Anwendungsprogramme
/proc	Systeminformationen
/root	Homeverzeichnis vom Root Benutzer



## Was ist wo? – Verzeichnisstruktur 2

/run	Relevante Daten für laufende Prozesse
/sbin	System Binaries, Programme für root (fsck, route, ifconfig, ...)
/srv	Daten für Dienste (z.B. Webserver, FTP-Server, ...)
/sys	Information die von Treibern oder vom Kernel bereitgestellt werden
/tmp	Temporäre Dateien
/usr	Softwarepakete des Herstellers
/usr/local	Lokal installierte Software, nicht Bestandteil der Distribution
.../bin	ausführbare Programme
.../include	Header-Dateien
.../lib	Bibliotheken
.../share	Daten wie Icons, Sounds
.../src	Source Code
/var	Variable Daten (Logs, Emails, Warteschlange)

## Wildcards

- Wildcards werden von der Shell aufgelöst (Pattern-Matching)
- Details hängen von der Shell ab (man bash!)
- „\*“ passt auf jeden String, auch auf den leeren
- „?“ passt auf genau ein Zeichen
- „[a-zA-Z]“ passt genau auf die angegebenen Zeichen(bereiche)

- Beispiele:

Pattern	Treffer
*	a abcd a.txt b bc d d3
*.*	a.txt
*b?	bc
*d*	abcd d d3
[a-z][0-9]	d3

Inhalt des Verzeichnis:

a  
 abcd  
 a.txt  
 b  
 bc  
 .d  
 d  
 d3

## Erweitertes Suchen von Dateien: find

- `find <path>... <expression>`
- Suchen von Dateien und Verzeichnissen
- Viele Kriterien, die auch noch verknüpft werden können  
⇒ man find

### Beispiele

- `find . -size +500c`  
⇒ sucht Dateien mit >500 Zeichen
- `find . -name "*.txt" -a -mtime -1`  
⇒ sucht .txt-Dateien, die vor <24h geändert wurden
- `find . -type d`  
⇒ findet alle Unterverzeichnisse (Typ d wie directory)
- `find ~ ! -type d`  
⇒ findet alle Nichtverzeichnisse im Home

## grep: Suchen in Dateien

grep [-rin] <string> <file>...

- Sucht nach String oder Muster in Textdateien
- -r: Auch in Dateien in Unterverzeichnissen
- -i: Groß-/Kleinschreibung ignorieren
- -n: Gibt die Zeilennummer mit aus
- Mächtigeres Tool: awk

## Beispiele

grep -i linux \*.txt

⇒ Durchsucht alle .txt-Dateien nach "linux", "Linux", "LINUX" und so weiter

grep -r Linux .

⇒ Durchsucht alle Dateien in allen Unterverzeichnissen

find . -name "\*.txt" -exec grep Linux {} \;

⇒ Durchsucht nur .txt-Dateien

## Tail / Head – Anfang oder Ende einer Textdatei

tail [-fn] <file>

- Zeigt das Ende einer Textdatei (neuste Einträge)
- Ohne Parameter werden die letzten 10 Zeilen angezeigt
- -f = follow: Programm beendet sich nicht und zeigt neu geschriebene Einträge an  
TIPP: Ideal für Logdateienbetrachten
- -n [x]: Die gewünschten letzten x-Zeilen der Datei ausgeben

head [-n] <file>

- Zeigt den Anfang einer Textdatei
- Ohne Parameter werden die ersten 10 Zeilen angezeigt
- -n [x]: Die gewünschten ersten x-Zeilen der Datei ausgeben

## Benutzer

- Jeder Prozess und jede Datei gehört einem Benutzer
- Jeder Benutzer gehört einer oder mehreren Gruppen an
- Anmeldung meist mit Passwort
- Benutzer root hat immer alle Rechte  
⇒ nie als root arbeiten!

w, who	Wer ist gerade angemeldet?
whoami	Wer bin ich gerade?
groups	Meine Gruppen anzeigen
id	Meine UserID anzeigen
passwd	Passwort ändern
su <user>	Benutzer wechseln
sudo <command>	Befehl als root ausführen, wenn erlaubt

## Datei- und Verzeichnisrechte

ls -l <file>	Dateirechte ausgeben
chmod ugoa[= + -]rwx <file>	Dateirechte ändern
chown <user>:<group> <file>	Besitzer/-gruppe ändern

- Jede Datei gehört einem Benutzer und einer Gruppe
- Rechte für Benutzer (u), Gruppe (g) und andere (o), oder alle (a)
- Kürzel für die meist benutzten Rechte:

Kürzel	bei Dateien	bei Verzeichnissen
r	Lesen	Dateien anzeigen
w	Schreiben	Dateien anlegen/löschen
x	Ausführen	In das Verzeichnis wechseln

- Feinere Rechte durch ACLs (Zugriffskontrolllisten)



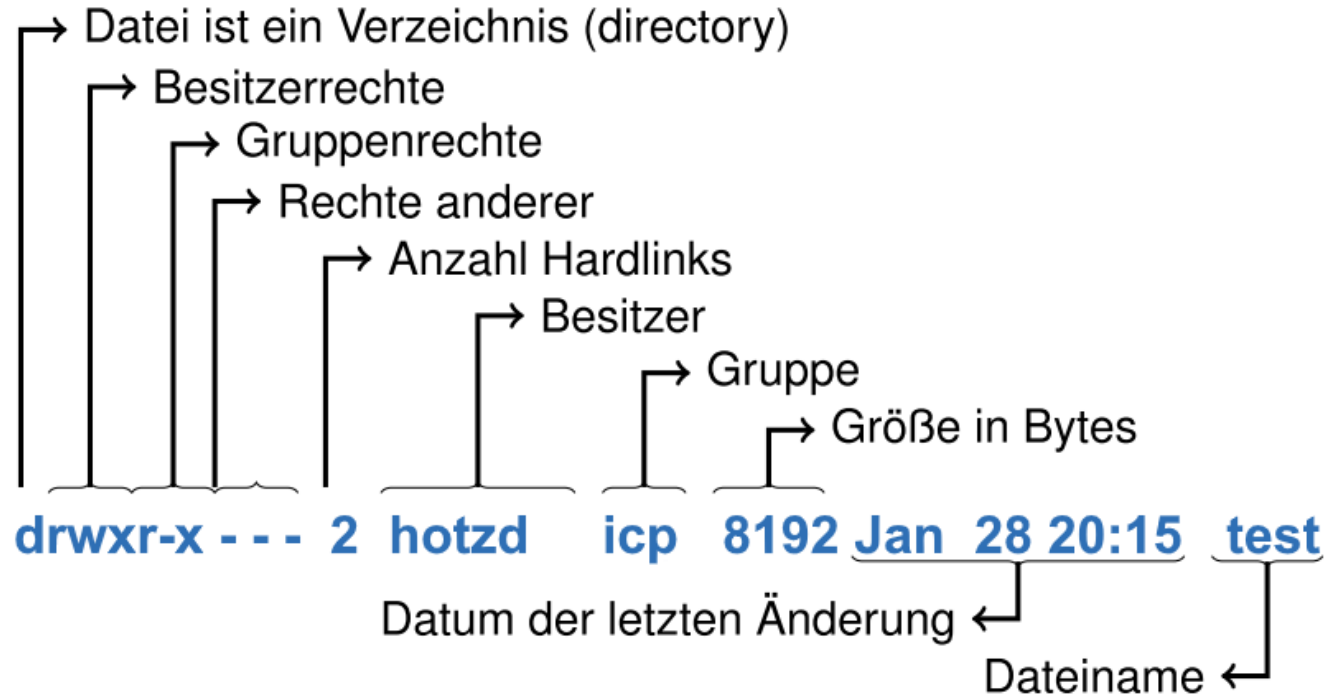
# Datei- und Verzeichnisrechte

Oktal	Binär	Kürzel	Menschlich
0	000		nichts
1	001	x	ausführen
2	010	w	schreiben
3	011	wx	schreiben, ausführen
4	100	r	lesen
5	101	rx	lesen, ausführen
6	110	rw	lesen, schreiben
7	111	rwX	lesen, schreiben, ausführen

- `chmod 755 <file>`

## Beispiele: Ausgabe von ls

`ls -ld test` gibt aus:



## Beispiele: Dateirechte

- `mkdir test; chmod ug+rwx test`  
Legt ein Verzeichnis für die ganze Gruppe an
- `chmod u-rwx test`  
Jetzt darf ich selber nichts mehr in test:  
`mkdir test/bla`  
`bash: test/bla: Permission denied`
- `chmod og-rwx ~/Documents`  
„Documents“ vor allen anderen (außer root) verstecken
- `chmod a+rx ~/bin/superprog`  
Das Programm „superprog“ für alle ausführbar machen
- `chmod -R 777 *`  
Alle Dateien im aktuellen Arbeitsverzeichnis inkl. alle Unterverzeichnisse bekommen voll Berechtigungen, sollte aus Sicherheitsgründen auf einem Mehrbenutzersystem vermieden werden

## Prozesse

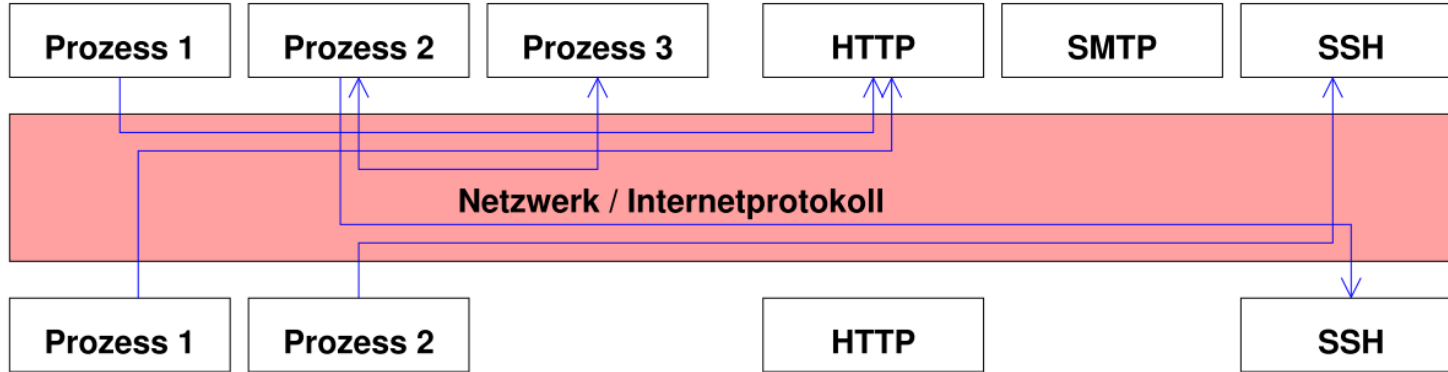
- Multitasking - (fast) beliebig viele Prozesse „gleichzeitig“
- Prozess: Eine laufende Instanz eines Programmes
- Control-z stoppt den aktuell laufenden Prozess
- Ein Befehl, der mit „&“ endet, wird im Hintergrund gestartet
- Ein Hintergrundprozess erhält keinen Tastaturinput, aber schreibt auf das Terminal
- Grundlegende Prozessbefehle:

ps	Anzeige der aktuell laufenden Prozesse
top	Fortlaufende Anzeige der aktivsten Prozesse
bg	Gestoppten Prozess in den Hintergrund
fg	... und wieder in den Vordergrund
kill <pid>	Prozess beenden
killall <name>	Prozesse mit diesem Namen beenden

# Netzwerk

## Anwenderprozesse

## Dienste (Daemonen)



- Wir beschränken uns auf das Internet-Protokoll
- IP-Adresse oder DNS-(Nameservice-)Name bestimmen Rechner
- Dienstprogramme warten auf Ports auf Anfragen
- Ports haben Nummern, Dienste zugeordnete Portnummern  
 (http=80, ssh=22,...)

## Beispiele von Netzwerkdiensten

Protokoll	Server	Client
http (Hypertext, WWW)	Apache, lighttpd, nginx	Firefox, Chrome, Edge
smtp	sendmail, postfix, exim	Thunderbird, Kmail, Evolution, Outlook
pop3	courier, cyrus, dovecot	
imap		
ftp (File Transfer)	vsftpd, proftpd, filezilla	ftp, ncftp, firefox, chrome
ssh (Secure Shell)	sshd	ssh, putty
cups (Druckerspooles)	cupsd	lpr, Programme mit Druckerdialog
X11 (GUI)	Xorg, FX86	Alle Programme mit GUI
NFS (Network File System)	nfsd, Linux-Kernel	nfs, bestandteil des Linux Kernels

## SSH: Secure Shell

<code>ssh [-X] [&lt;user&gt;@]&lt;host&gt;</code>	Terminal für Benutzer auf einem anderen Rechner öffnen
<code>scp &lt;src&gt; &lt;host&gt;:&lt;dst&gt;</code>	Eine Datei per SSH auf einen anderen Rechner kopieren

- Verschlüsselte Verbindung zu einem anderen Rechner
- Terminal, Dateitransfer, Umleiten von Netzwerkverkehr
- Mit -X: Umleiten von X11, GUI-Programme per Netzwerk starten
- Client gehört auf allen UNICES zum Standard
- Windows-Client: PuTTY

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>



## Netzwerkbefehle unter Unix/Linux

<code>nc &lt;host&gt; &lt;port&gt;</code>	Verbindung zu einem Port aufbauen
<code>nc -l -p &lt;port&gt;</code>	Als Dienst auf einem Port lauschen
<code>hostname</code>	Wie heißt mein Rechner?
<code>netstat</code>	Zeigt aktive Ports
<code>ifconfig</code>	Infos über Netzwerkkarten (IP- und MAC-Adressen!)

### Ein paar Tipps

- Eine IPv4-Adresse hat die Form a.b.c.d mit  $0 \leq a,b,c,d \leq 255$
- Eigener Rechner ist stets auch „localhost“ (127.0.0.1)
- Benutzergestartete Dienste nur auf Ports > 1024
- Per NFS können Verzeichnisse auf mehreren Rechnern gleichzeitig zu sehen sein

## Skripte

- Wie kann ich mir komplexe Befehle merken?
- Gar nicht – aber der Computer kann es für mich!
- Die Befehle in eine Textdatei schreiben und ausführbar machen
- „#!“ (Shebang oder Hash-Bang) in der ersten Zeile bestimmt ausführende Shell
- Kommandozeilenwerte als „\$1“, „\$2“, ...

## Beispiel

- Datei mvtonewdir.sh erstellen mit Inhalt:  
    #!/bin/bash  
    mkdir -p \$2 && chmod og-rwx \$2 && mv \$1 \$2
- `chmod a+rx mvtonewdir.sh`  
    ausführbar machen
- `./mvtonewdir.sh bla test`  
    und starten

## /etc/shadow

- \$1 = MD5
- \$2a =Blowfish
- \$2y=eksblowfish
- \$5 =SHA-256
- \$6 =SHA-512
- \$y =yescrypt

pi:\$6\$w3kkkfaMvVrL9g1e\$2ppkOxkHiQ.j72CUv.27p8bJbjHnBqjGKs.:18188:0:99999:7:::

- The local username
- The password hash
- Number of days since the start of unix time (01/01/1970) that the password was last changed
- Minimum number of days before the password can be changed
- Maximum number of days before the password must be changed. 99999 means that the user will not be forced to change their password
- Number of days before forcing the password change that the user will be warned.
- The number of days after expiration that the account will be disabled
- Days since the start of unix time that the account has been disabled
- Currently unused but reserved for future use

## Ein- und Ausgabe

- Ein Prozess hat wenigstens drei Dateien:

0	stdin	Eingabe (etwa Tastatur)
1	stdout	Standard-Ausgabe
2	stderr	Fehler-Ausgabe

- Wir können diese unabhängig umleiten:

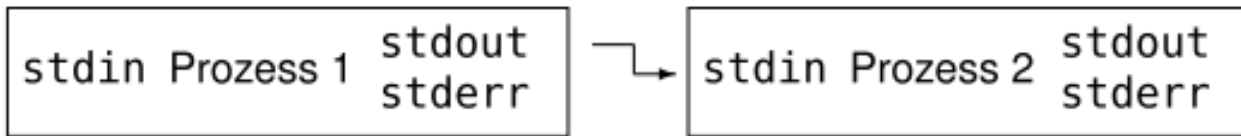
> <file>	Umleiten von stdout in eine neue Datei
>> <file>	Wie >, aber hängt an eine vorhanden Datei
>&	Umleiten von stderr & stdout
< <file>	Liest Datei als stdin

- Reihenfolge ist wichtig!
- Praktisch: Datei /dev/zero dient als leere Eingabe, /dev/null, um Ausgabe zu verschlucken

## Beispiele: Umleiten von Ein- und Ausgabe

- `grep Hase *.txt > hasen`  
Kopiert alle Zeilen, die „Hase“ enthalten, in Datei „hasen“
- `grep Igel *.txt >> hasen`  
Fügt alle Zeilen, die „Igel“ enthalten, an
- `ls *.txt >& errors > txt`  
Listet alle .txt-Dateien in Datei „txt“, Fehler in Datei „errors“
- `ls *.txt > txt >& errors`  
(!) Datei „txt“ ist leer, Fehler und .txt-Dateien in „errors“
- `grep Igel < hasen`  
Gibt alle Zeilen der Datei „hasen“ aus, die Igel enthalten
- `./myprogram < /dev/zero >& stderr > /dev/null &`  
Startet „myprogram“ im Hintergrund ohne Ein- oder Ausgabe, nur Fehler werden in „stderr“ gespeichert

## Verknüpfen von Prozessen: Pipes



- „|“ verbindet Ausgabe eines Prozesses mit Eingabe eines anderen
- Mit „;“ können zwei Prozesse nacheinander gestartet werden

## Beispiele

- `cd bla; ls`  
In Verzeichnis „bla“ wechseln und Dateien darin ausgeben
- `ps axww | grep bash | grep -v grep`  
Sucht alle laufenden bash-Shells, ohne den grep-Befehl auszugeben
- `ls -a | grep txt`  
Listet alle Dateien mit „txt“ im Namen

## Rückgabewerte

- Wie kann ich auf ein Ergebnis reagieren?
- Prozesse Rückgabewert
- Meist 0: Erfolg, >0: Fehler, etwas ist schief gegangen
- Wert kann durch „\$?“ in der Shell abgefragt werden
- `proc1 && proc2`  
proc2 startet genau dann, wenn proc1 keinen Fehler meldet
- `proc1 || proc2`  
proc2 startet genau dann, wenn proc1 einen Fehler meldet
- Wert kann bei **exit** gesetzt werden

## Beispiele

- `ls *.txt || echo "Keine .txt-Datei hier"`
- `test -f "bla" && cat bla`  
Gibt die Datei „bla“ nur aus, wenn sie existiert



## Shell-Variablen

- In der Shell kann man Werte mit „=“ speichern und mit „\$“ auslesen

name=bla; echo "name ist \$name"

name ist bla

- Leerzeichen trennen Befehle, müssen daher in Hochkommata

**Falsch:** i=Meine Fotos

Fotos: command not found

⇒ Setzt i=Meine und führt dann Befehl Fotos aus

**Richtig:** i="Meine Fotos"

- es darf kein Leerzeichen vor oder nach „=“

**Falsch:** i = 1

i: command not found

⇒ Führt Befehl i mit Parametern = und 1 aus

**Richtig:** i=1

## Shell-Variablen

- Skript-Parameter verfügbar als \$1, \$2, ...
- \$0 ist Skriptname
- \$\* ist Liste alle Parameter

```
echo "usage: $0 <file> <dir>, not $*"
usage: mvtonewdir.sh <file> <dir>, not a b c
```
- \$? gibt den Rückgabewert des letzten Befehls
- Ausgabe eines Befehls per „ ` “ (Backtick) als String

```
files= `find . -name "*.txt"`; echo $files
./text1.txt ./text2.txt
```

⇒ Variable files enthält die Standardausgabe des find-Befehls

Tipp: Backtick nützlich z.B. wenn man Dateinamen mit den aktuellen Datum/Uhrzeit generieren möchte ``date +%Y%m%d``

## Musterersetzung

<code>\${&lt;var&gt;%&lt;pat&gt;}</code>	pat am Ende entfernen
<code>\${&lt;var&gt;%%&lt;pat&gt;}</code>	wie oben, aber alle Treffer
<code>\${&lt;var&gt;#&lt;pat&gt;}</code>	pat am Anfang entfernen
<code>\${&lt;var&gt;##&lt;pat&gt;}</code>	wie oben, aber alle Treffer
<code>\${&lt;var&gt;//&lt;abc&gt;/&lt;def&gt;}</code>	abc durch def ersetzen

### ■ Beispiele (TEST=/home/hotz/abc.txt.old)

`${TEST%.old}` → /home/hotz/abc.txt

`${TEST%.*}` → /home/hotz/abc

`${TEST%%.*}` → /home/hotz/abc

`${TEST#*/}` → home/hotz/abc.txt.old

`${TEST##*/}` → abc.txt.old

`${TEST//ab/de}` → /home/hotz/dec.txt.old

(alles hinter dem letzten Punkt abschneiden)

(alles hinter dem ersten Punkt abschneiden)

(alles vor dem erste / entfernen)

(alles vor dem letzten / entfernen)

(suchen ersetzten, da verdoppelt, alle Treffer)

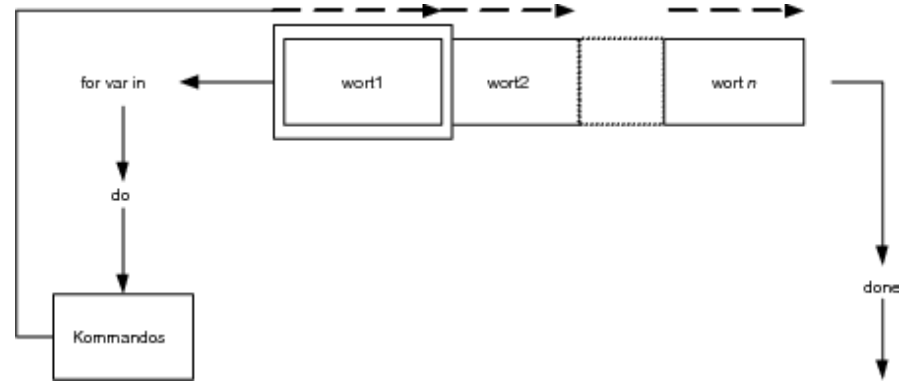
## Schleifen - for

```
for var in liste_von_parameter
do
    Befehl
    ...
    Befehl
done
```

### Beispiel:

```
for var in wort1 wort2 wort3
do
    echo $var
done
```

- `for var in "$@"` übernimmt alle Parameter des Skriptes
- `for datei in *` für die Schleife für jede gefundenen Datei im Arbeitsverzeichnis



## Schleifen - for

### Mehr Beispiele:

- `for f in 1 2 3 4; do echo -n "$f,"; done;`  
`1,2,3,4,`

- `for f in *.txt; do`  
`n=${f%.txt}-2.txt; mv $f $n`  
`done`

⇒ Benennt alle Textdateien um und gibt aus, was es tut

## Schleifen - while

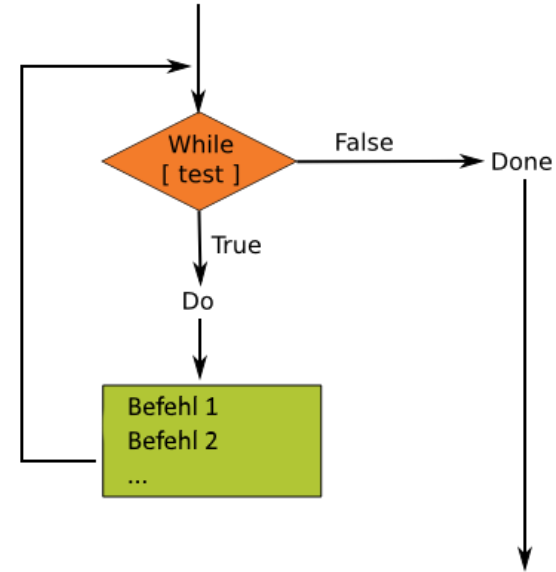
```
while [bedingung]
do
  Befehl 1
  ...
  Befehl n
done
```

### Beispiel

```
#!/bin/bash
COUNTER=0
while [ $COUNTER -lt 10 ]; do
  echo The counter is $COUNTER
  let COUNTER=COUNTER+1
done
```

alternative ((COUNTER++))

- Tipp: Einzeiler Endlos-skript: `while true; do [Befehl]; sleep 1; done`

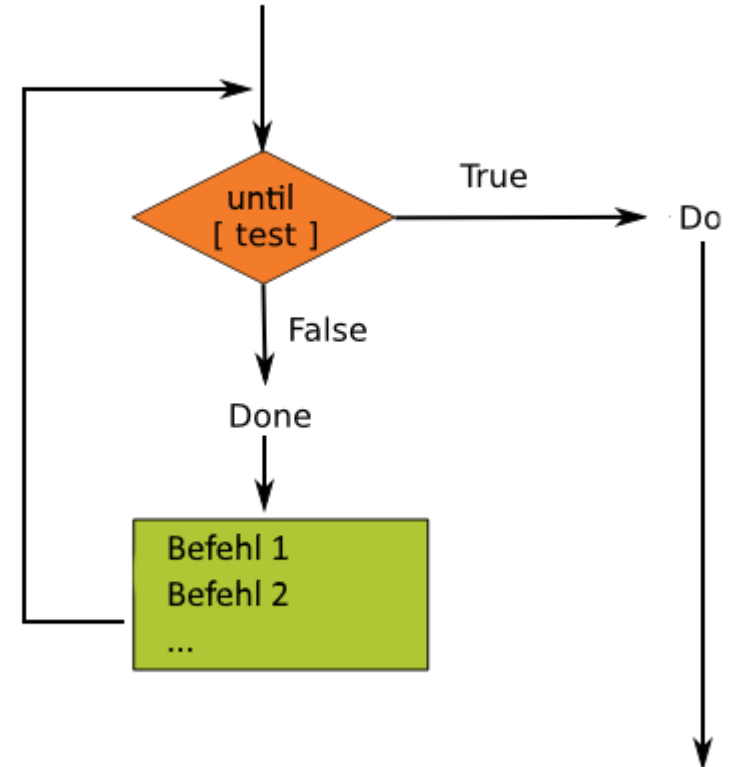


## Schleifen: until

- Wie die while Schleife, nur läuft die Schleife solange bis ungleich 0 false zurückgeliefert wird

### Beispiel:

```
#!/bin/bash  
COUNTER=20  
until [ $COUNTER -lt 10 ]; do  
    echo COUNTER $COUNTER  
    let COUNTER-=1  
done
```



## Bedingte Ausführung - if

```
if [!] cond; then cmd1; [ else cmd2; ] fi
```



- ! Dreht die Bedingung genau um
- Das test kann auch durch [ ] ersetzt werden, bitte auf Leerzeichen achten

### Beispiele

- if test -f \$f; then echo "\$f gibt es"; fi  
 oder  
 if [ -f \$f ]; then echo "\$f gibt es"; fi  
 Test, ob \$f eine Datei ist
- if ! test -f \$f; then echo "\$f fehlt"; fi  
 Gibt nur etwas aus, wenn es \$f nicht gibt
- if grep -q bla \$f; then echo "bla in \$f"; fi  
 Meldet, ob \$f „bla“ enthält



## Vergleiche: test

- "`<string1>`" = "`<string2>`": Zeichenketten gleich
- "`<string1>`" != "`<string2>`": Zeichenketten ungleich
- `<zahl1>` -eq `<zahl2>`: Zahlenwert gleich
- `<zahl1>` -gt/-ge `<zahl2>`: Zahlenwert größer (gleich)
- `<zahl1>` -lt/-le `<zahl2>`: Zahlenwert kleiner (gleich)
- -f "`<datei>`": ist reguläre Datei
- -d "`<datei>`": ist Verzeichnis
- `<bedingung1>` -o `<bedingung2>`: logisch oder
- `<bedingung1>` -a `<bedingung2>`: logisch und


## Beispiele

- `test "$a" -ge "$b" -a "$a" -le "$b"`

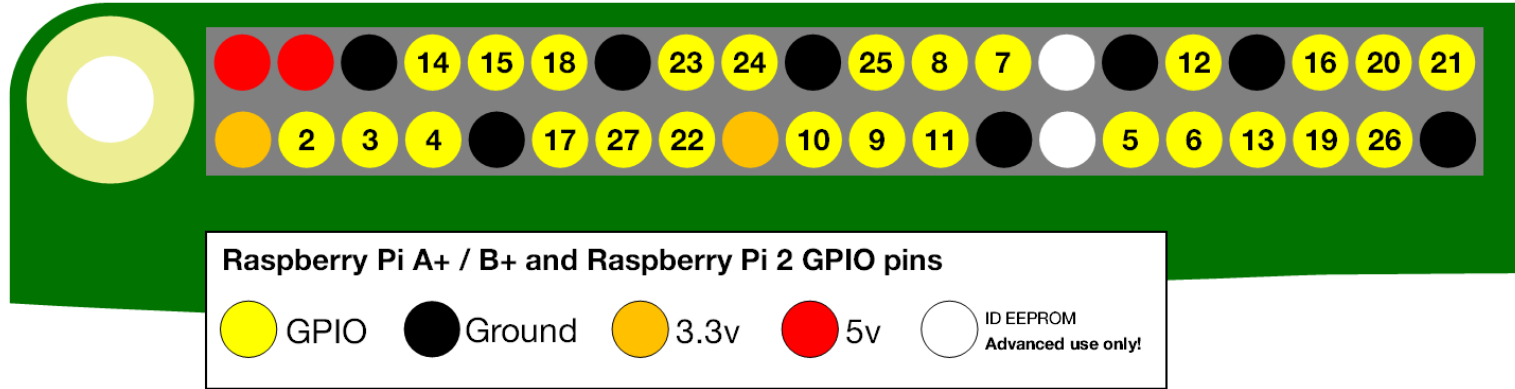
Komplizierte Variante von `$a -eq $b`

- `if [ $a -eq 0 ]; then echo "Ist null!"; fi`

## Shellexpansion / -ersetzung

- Passiert bei Ausführung einer Zeile
- Tildeexpansion (außerhalb Anführungszeichen):  
    ~/Documents ⇒ /home/hotz/Documents
- Variablenexpansion (in doppelten Anführungszeichen):  
    echo "Hostname: \$HOSTNAME" ⇒ **Hostname: raspi**  
    **aber** echo 'Hostname: \$HOSTNAME' ⇒ **Hostname: \$HOSTNAME**
- Ersetzen von Backticks echo `date +%Y%m%d` ⇒ **20190309**
- Worteinteilung:  
  
ls /home/pi/\* "Hallo Welt"
- Wildcardersetzung:  
    echo ls /home/pi/\* "Hallo Welt"  
    **ls /home/pi/Desktop /home/pi/Documents /home/pi/Downloads Hallo Welt**

# Raspberry Pinout



- PWM (pulse-width modulation)
  - Software PWM available on all pins
  - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- SPI
  - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
  - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- I2C
  - Data: (GPIO2); Clock (GPIO3)
  - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
- Serial
  - TX (GPIO14); RX (GPIO15)

# Pinout

- Auf dem Raspberry im Terminal einfach `pinout` eingeben

```
root@raspistrom:~# pinout
-----
|oooooooooooooooooooo J8 | +===+
|looooooooooooooooooooo | USB | +===+
|                           |   | | | | |
| Pi Model 3B V1.2         |   |
| +-----+               |   |
| |D| |SoC|               |   |
| |S| |   |               |   |
| |I| +-----+           |   |
|                           |   |
|                           |C| +=====+
|                           |S| | Net  | +=====+
| pwr |                   |I| |A|   |
| |   |                   |V|   |
|-----|
Revision      : a02082
SoC           : BCM2837
RAM           : 1024Mb
Storage       : MicroSD
USB ports     : 4 (excluding power)
Ethernet ports : 1
Wi-fi         : True
Bluetooth     : True
Camera ports (CSI) : 1
Display ports (DSI) : 1
```

```
J8:
 3V3 (1) (2) 5V
GPIO2 (3) (4) 5V
GPIO3 (5) (6) GND
GPIO4 (7) (8) GPIO14
GND (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
 3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
GND (25) (26) GPIO7
GPIO0 (27) (28) GPIO1
GPIO5 (29) (30) GND
GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
GND (39) (40) GPIO21

For further information, please refer to https://pinout.xyz/
```

## GPIO – *General Purpose Input/Output*

- Der Raspberry hat 26 GPIO Ports
- `echo "[GPIONR]" > /sys/class/gpio/export`

### Aufgabe 2

- Bringen Sie die LEDs der Ampel zum Leuchten

## Aufgabe 2 Lösung

- `echo "16" > /sys/class/gpio/export`

Dies aktiviert den GPIO16 (Pin36) als Standard I/O.

- `ls /sys/class/gpio/`

`export gpio16 gpiochip0 gpiochip128 unexport`

Es ist ein neues Verzeichnis `gpio16` aufgetaucht.

- `echo "out" > /sys/class/gpio/gpio16/direction`

Man definiert diesen Pin als Ausgang.

- `echo "1" > /sys/class/gpio/gpio16/value`

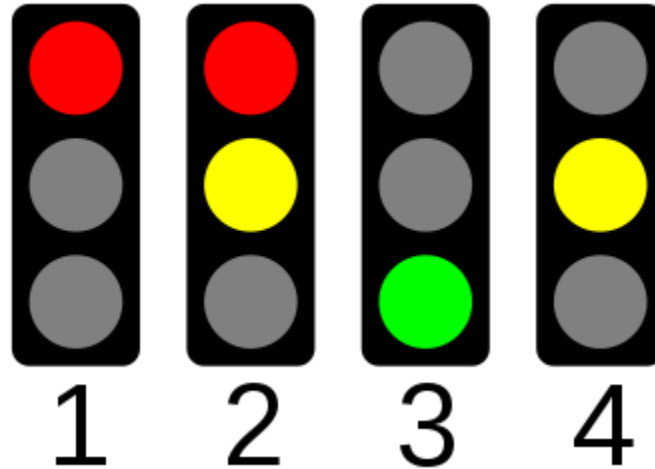
Die rote LED leuchtet.

Wiederholen für:

- GPIO20 – LED gelb
- GPIO21 – LED grün

## Aufgabe 3

Schreiben sie ein kleines Skript, das eine Ampel simuliert. Das Skript soll endlos laufen, bis [CTRL]+[C] gedrückt wird.



## Aufgabe 3 Lösung

```
#!/bin/bash

rot=16
gelb=20
gruen=21
zeit=3

#Init GPIO
for f in $rot $gelb $gruen; do
    echo $f > /sys/class/gpio/export
    sudo chmod 766 "/sys/class/gpio/gpio$f/direction"
    sudo chmod 766 "/sys/class/gpio/gpio$f/value"
    echo "out" > "/sys/class/gpio/gpio$f/direction"
done

while true; do
    echo 1 > "/sys/class/gpio/gpio$rot/value"
    sleep $zeit
    echo 1 > "/sys/class/gpio/gpio$gelb/value"
    sleep $(( $zeit/3 ))
    echo 0 > "/sys/class/gpio/gpio$rot/value"
    echo 0 > "/sys/class/gpio/gpio$gelb/value"
    echo 1 > "/sys/class/gpio/gpio$gruen/value"
    sleep $zeit
    echo 0 > "/sys/class/gpio/gpio$gruen/value"
    echo 1 > "/sys/class/gpio/gpio$gelb/value"
    sleep $(( $zeit/3 ))
    echo 0 > "/sys/class/gpio/gpio$gelb/value"
done
```



## Aufgabe 4

- Benutzen Sie den Taster des Drehimpulsgeber um mindestens eine LED zu steuern
- Der Taster ist am GPIO 13 angeschlossen

## Aufgabe 4 Lösung

```
while true; do cat /sys/class/gpio/gpio13/value > /sys/class/gpio/gpio16/value; sleep 0.01; done
```

Programmiertechnisch sehr schlecht, da ohne das Sleep eine Core zu 100% ausgelastet wird.

## Woher kommen die Programme?

### PATH: Befehlspfad

- Variable
- : getrennte Liste der Verzeichnisse, die Programme enthalten
- Sollte aus Sicherheitsgründen niemals . enthalten
- Vom System vorbereitet, vom Benutzer ergänzt
- Beispiel:

```
PATH="~/bin:$PATH"; echo $PATH  
/home/pi/bin:/usr/local/bin:/usr/bin:/bin
```

### LD\_LIBRARY\_PATH: Bibliothekspfad

- Dasselbe für dynamische (shared) Bibliotheken (.so-Dateien)
- Oft bei selbst kompilierter Software nötig

## Dynamische Bibliotheken

### Gemeinsamer Code vieler Programme

- Algorithmen: Sortieren, Speicherverwaltung
- Ein-/Ausgabe
- Graphisches Interface
- Codierung / Decodierung

### Dynamische Bibliotheken

- Von vielen Programmen gemeinsam benutzter Code
- Zur Laufzeit bei Bedarf geladen
- Sparen Festplatten- und Hauptspeicher
- Können geupdated werden, ohne alle Programme upzudaten
- ldd zeigt benötigte Bibliotheken

## Umgebungsvariablen

<code>export &lt;var&gt;[=&lt;value&gt;]</code>	Variable exportieren
<code>unset &lt;var&gt;</code>	Variable löschen

- Exportierte Variablen (Umgebungsvariablen) stehen allen aufgerufenen Programmen zur Verfügung
- Einige oft benutzte Umgebungsvariablen:

PWD	aktuelles Verzeichnis
HOME	Pfad zum Benutzerverzeichnis
DISPLAY	X-Server (meist lokal, „:0.0“)
PS1	Prompt (z.B. „\u:W“, Benutzer + Verzeichnis)
EDITOR	Standard-Texteditor (meist vim)
LANG	Spracheinstellung („de_DE“ oder „en_US.UTF-8“)

## Aliases: Abkürzungen

<code>alias &lt;short&gt;='&lt;cmd&gt;' ...</code>	Alias setzen
<code>unalias &lt;short&gt;</code>	Alias löschen
<code>alias</code>	Alle Aliases ausgeben

- Aliases erlauben verkürzte Befehlseingabe
- auch, um Befehlen Default-Parameter zu geben

### Beispiele

- `alias cp='cp -i'      mv='mv -i'      rm='rm -i'`  
Löschen und überschreiben nur auf Nachfrage (!)
- `alias ls='ls --color=auto'`  
Dateien bei Ausgabe nach Typ einfärben
- `alias ll='ls -l'`  
Ausführliche Ausgabe

## Konfigurationsdateien

- Konfiguration von UNIX-/GNU-Programmen über Textdateien
- Versteckte Dateien im Home-Directory
- Oft `.<program>rc` (rc für Resource)
- Beispiele:

<code>.bashrc</code> <code>.profile</code>	wird von der Shell beim Start/Login gelesen Setzt Variablen (Pfad), aliases, ... <b>Achtung, Fehler hier drin können es unmöglich machen, sich anzumelden!</b> Immer erst per <code>source .bashrc</code> testen
<code>.ssh/config</code>	Einstellungen für ssh, z.B. <code>ForwardX11 Yes</code>
<code>.vimrc</code>	Einstellungen für vim, z.B. <code>syntax on</code>
<code>.selected_editor</code>	Ausgewählter default Texteditor

## Nützliche Tools (1)

- Datum: `date`
- Leere Datei erzeugen: `touch <datei>`
- Archivieren: `tar <optionen> <dateien>`
  - `cf <Archiv>`: Archiv erzeugen
  - `xf <Archiv>`: Archiv wieder auspacken
  - `z`: gzip-Packen des Archives (`.tar.gz` / `.tgz`) oder
  - `j`: bzip2-Packen des Archives (`.tar.bz2`)
  - `J`: xz-Packen des Archives (`.tar.xz`)
  - `v`: mehr Ausgabe
- Bilder skalieren, konvertieren, ...:  
`convert <quelle> <optionen> <ziel>`  
Bestandteil des Paketes `imagemagick` zum installieren: `sudo apt install imagemagick`
- Bash Berechnungen durchführen  
`bc -l`  
Beispiele: `bc -l <<< 5*5` oder `echo "5*5" |bc`



## Beispiel: einfaches Backup

```
#!/bin/bash
# sichert Home-Verzeichnis in Unterverzeichnis
# dieses wird bei Bedarf erzeugt
# ... aber natürlich nicht mit gespeichert
# Verzeichnis im Home
BACKUPDIR=Backup
# wie das Backup heißen soll
DATE=`date "+%Y-%m-%d"`
NAME="$BACKUPDIR/backup-$DATE.tar.gz"
cd ~
# wenn das Verzeichnis nicht existiert, erzeugen
mkdir -p "$BACKUPDIR"
# und sichern
tar --exclude=$BACKUPDIR -vczf "$NAME" ~
```

## Nützliche Tools (2)

- `sleep 60`  
Wartet 60 Sekunden (verbraucht keine System-Ressourcen)
- `wc report.txt` (word count)  
`438 2115 18302 report.txt`  
Zählt die Anzahl Zeilen, Worte und Zeichen in einer Datei oder der Standardeingabe.

## awk: Zum zeilenweise Auswerten oder Editieren

awk [-F wert] Bedingung { Anweisungen }

- -F: field separator, Trennzeichen standardmäßig Leerzeichen
- Ist selbst eine Skriptprogrammiersprache

### Beispiele

awk '\$2 == "localhost" { print \$1 }' /etc/hosts

⇒ Durchsucht die hosts Datei und gibt das erste Feld von einer Zeile aus, wenn im zweite Feld localhost steht

cat /proc/cpuinfo |grep Serial |awk '{print \$3}'

⇒ Gibt die Seriennummer des Raspberries aus

cat /etc/passwd |grep pi |awk -F : '{ print \$1 \$3 }'

⇒ Gibt den Username pi zusammen mit der UID aus

## cut: Zum zeilenweise Extraktion gewisser Spalten

cut OPTION... [FILE]...

- -b: Byte
- -c: Zeichen (character)
- -f: Feld
- Extrahiert die entsprechenden Teile aus der Zeile
- -d Trennzeichen

### Beispiele

cut -c 18-25 /proc/meminfo

⇒ Gibt die Zeichen 18-25 jeder Zeile aus

cat /proc/cpuinfo |grep Serial |cut -d : -f 2

⇒ Gibt die Seriennummer des Raspberries aus

cat /etc/passwd |grep pi |cut -d : -f 3

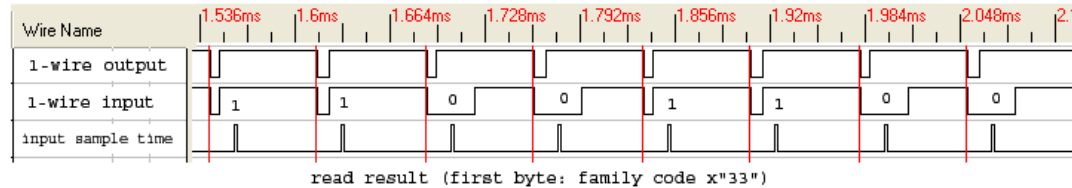
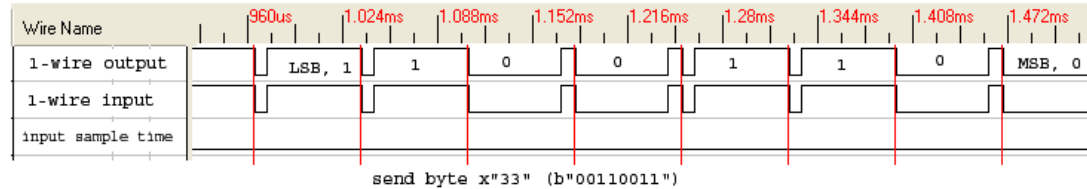
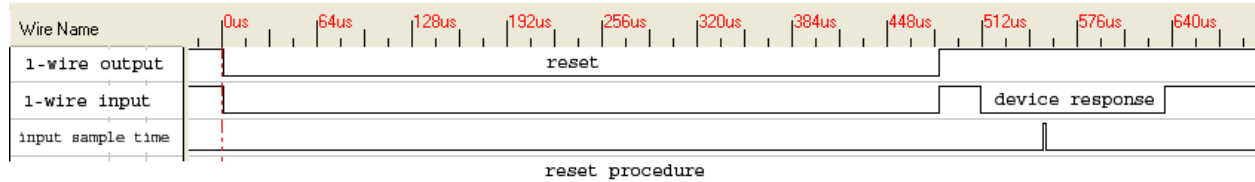
⇒ Gibt die UID vom User pi aus

## 1-wire

- Produkt der Firma Dallas Semiconductor Corp.
- Ist ein bidirektionales serielles Protokoll.
- Halbduplex - Asynchron
- Bis zu 100 Slaves an einem Bus, aber nur ein Master
- Adressierung 64-Bit-ROM-ID:  
8-Bit-Family-Code, 48-Bit-Seriennummer, (8-Bit-CRC-Prüfsumme)  
Beispiele: **28-000006880049**
- Stromversorgung erfolgt über die Datenleitung, die durch einen Kondensator gepuffert wird.

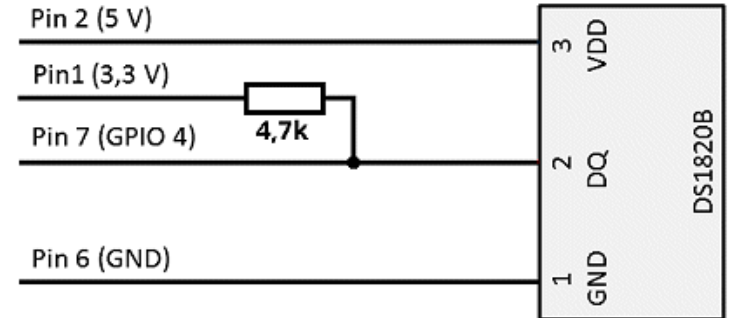
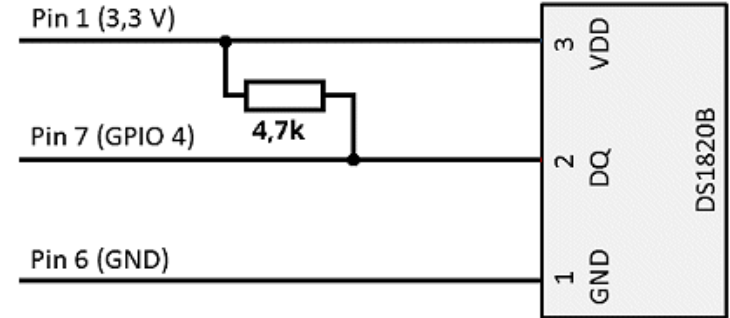
# Protokoll 1-wire

1 Wire reset, write and read example with DS2432



## DS18B20

- Temperatur Sensor
- $-55^{\circ}\text{C} - 125^{\circ}\text{C}$
- $\pm 0,5^{\circ}\text{C}$  Genauigkeit zwischen  $-10^{\circ}\text{C} - 85^{\circ}\text{C}$
- Auflösung programmierbar 9 Bits -12 Bits
- 1-wire



## Aufgabe 5

- Gebe die Temperatur aus, nur die ganze Zahl vor dem Komma.



## Aufgabe 5 - Lösung

- `cat /sys/bus/w1/devices/28-00000a936750/w1_slave |tail -n 1 |cut -d= -f2`
- `cat /sys/bus/w1/devices/28-00000a936750/w1_slave |grep t |sed 's/.*=//'`

## Sed – Stream Editor

- Zeilenweise Text-Datenstrom Editor
  - Anwendung in Skripten, nicht interaktiv wie normale Texteditoren
  - Suche laut der POSIX-Spezifikation eine bestimmte Abart von RegEXP
  - Sehr mächtig, daher hier nur ein paar einfache Beispiele, mehr Infos liefern die Man-Pages oder das Internet
- 
- `sed 's/regexp/replacement/g' inputFileName > outputFileName`
  - `sed -i 's/regexp/replacement/g' FileName`  
In diesem Beispiel ist FileName die Input/Output Datei
- 
- Beachte: Klammern müssen escaped werden `\( \) \{ \}`

## Sed – Suchen und Ersetzen

- sed 's/alt/NEU/' inputfile

Ersetzt den ersten gefunden Treffer

- Beispiel:

```
echo "alt alt alt alt" | sed 's/alt/NEU/'  
NEU alt alt alt
```

- sed 's/alt/NEU/g' inputfile

Ersetzt alle gefunden Treffer

- Beispiel:

```
echo "alt alt alt alt" | sed 's/alt/NEU/g'  
NEU NEU NEU NEU
```

## Sed - Parameter

Befehl	Bedeutung	Beispiel
p	Print - Gibt die entsprechenden Zeilen auf dem Bildschirm aus	sed -n '1,5p' Datei – zeigt nur die ersten fünf Zeilen an
d	Löscht den definierten Bereich	sed '4d' Datei – löscht die vierte Zeile
s	Ersetzt Zeichenketten	sed 's/alt/NEU/g' Datei – ersetzt alle Vorkommen von „alt“ durch „NEU“
a	Fügt Text hinter den adressierten Zeilen	1a\ "Neue Zeile" – fügt nach der ersten Zeile den nachfolgenden Text ein.
i	Fügt Text vor den adressierten Zeilen ein	1i\ "Neue Zeile" – fügt vor der ersten Zeile den nachfolgenden Text ein.
c	Ersetzt Zeilen oder Zeilenbereiche	2c\ "Neue Zeile" – ersetzt die zweite Zeile durch den folgenden Text
r	Liest den Inhalt einer Datei ein und setzt diesen hinter die entsprechende Adresse.	sed '2r neu.txt' Datei – setzt den Inhalt von neu.txt hinter die zweite Zeile von Datei.
w	Schreibt die adressierten Zeilen oder Bereiche in eine neue Datei.	sed '5,\$w neu.txt' Datei – speichert ab der fünften bis zum Ende alles in der Datei neu.txt

## Suchmuster – „regular expressions“

Muster	Bedeutung
abc	genau diese Zeichenkette: „abc“
[abc]	eines dieser Zeichen: a, b oder c
[^abc]	ein beliebiges Zeichen, außer a, b oder c
[a-c]	eines der Zeichen von a bis c
.	ein beliebiges Zeichen
?	das Muster vor dem ? darf einmal oder gar nicht auftreten, entspricht $\{0,1\}$
*	das Muster vor dem * darf beliebig oft oder gar nicht auftreten, entspricht $\{0,\}$
+	das Muster darf beliebig oft, muss aber mindestens einmal auftreten, entspricht $\{1,\}$
{n}	das Muster muss genau n-mal auftreten, entspricht $\{n,n\}$
{,n}	das Muster darf höchstens n-mal auftreten
{n,}	das Muster muss mindestens n-mal auftreten
{n,m}	das Muster muss mindestens n-mal und höchstens m-mal auftreten

## SED ausschneiden

- `sed 's/^([a-Z]*\) .*^1/' inputfile`

Zeigt das erste Wort der Zeile an

- Beispiel:

```
echo "eins zwei drei vier" | sed 's/[a-z]* [a-z]* \([a-z]*\) .*/NEU \1/'  
NEU drei
```

- Beispiel:

```
echo "eins zwei drei vier" | sed 's/[a-z]* [a-z]\{4\} \([a-z]*\) .*/NEU  
\1/'  
NEU drei
```

## Aufgabe 6

- Verwenden Sie sed um die Temperatur auszuschneiden und noch an der richtigen Stelle einen Punkt einzufügen
- Befehl `cat /sys/bus/w1/devices/28-00000a936750/w1_slave`

## Lösung 6

```
cat /sys/bus/w1/devices/28-9a4576126461/w1_slave | sed -n '2p' \
| sed 's/.* t=\([0-9]*\) \([0-9]\{3\}\)/Temperatur: \1.\2°C/'
```



## RRDtool – Round-Robin-Database

- Ist ein Tool für zeitbezogene Messdaten, zum Erfassen und Visualisieren
- Round-Robin (Rundlauf-Verfahren ~ Ringpuffer)
- Beim Erzeugen einer RRD Datenbank wird bereits der komplette Speicher allokiert
- Zum Installieren bitte folgenden Befehl eingeben:  
`sudo apt install rrdtool`

## rrdcreate

```
root@raspi:~# rrdcreate --help
```

RRDtool 1.6.0 Copyright by Tobi Oetiker

Usage: rrdcreate <filename>

    [--start|-b start time]

    [--step|-s step]

    [--no-overwrite]

    [DS:ds-name:DST:dst arguments]

    [RRA:CF:cf arguments]

DST: *GAUGE | COUNTER | DERIVE | DCOUNTER | DDERIVE | ABSOLUTE*

*Ausführliche Anleitung findet ihr hier:*

<https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html>

## Eine RRD Datenbank erzeugen

```
#!/bin/bash  
rrdtool create /home/pi/temperatur1.rrd --step 5 \  
DS:temp1:GAUGE:600:-30:100 \  
RRA:AVERAGE:0.5:1:17280 \  
RRA:AVERAGE:0.5:12:43200 \  
RRA:AVERAGE:0.5:120:52560 \  
RRA:AVERAGE:0.5:17280:7300 \  
RRA:MIN:0.5:17280:7300 \  
RRA:MAX:0.5:17280:7300
```

```
#!/bin/bash  
rrdtool create /home/hotz/temperatur2.rrd --step 5 \  
DS:temp1:GAUGE:600:-30:100 \  
RRA:AVERAGE:0.5:5s:1d \  
RRA:AVERAGE:0.5:1m:30d \  
RRA:AVERAGE:0.5:10m:1y \  
RRA:AVERAGE:0.5:1d:20y \  
RRA:MIN:0.5:5s:1d \  
RRA:MAX:0.5:5s:1d
```

## RRD Datenbank mit Daten füttern

- RRDtool

`rrdupdate --help`

RRDtool 1.6.0 Copyright by Tobi Oetiker

Usage: `rrdupdate <filename>`

`[--template|-t ds-name[:ds-name]...]`

`[--skip-past-updates]`

`time|N:value[:value...]`

`at-time@value[:value...]`

`[ time:value[:value...] ..]`

- `rrdupdate /home/pi/temperatur1.rrd N:$value`

## Aufgabe 7

- Bitte erstellt ein Skript, das endlos läuft und den Temperaturwert des DS18B20 regelmäßig (alle 5s) in die zuvor erzeugte RRD Datenbank abspeichert.

# Lösung 7

```
#!/bin/bash

slave_address="28-00000a936750"
filename="/home/pi/temperatur1.rrd"

while true
do
    value=`cat /sys/bus/w1/devices/$slave_address/w1_slave | tail -n 1 | sed 's/.*t=\([0-9]\{2\}\)\([0-9]\{3\}\)/\1.\2/'`
    echo `date +%H:%M:%S` Temperatur ist $value °C
    rrdupdate $filename N:$value
    sleep 4
done
```

# rrdgraph

- Usage: rrdtool [options] command command\_options
- \* graph - generate a graph from one or several RRD
- rrdtool graph filename [-s|--start seconds] [-e|--end seconds]
- [-x|--x-grid x-axis grid and label]
- [-Y|--alt-y-grid] [--full-size-mode]
- [--left-axis-format format]
- [-y|--y-grid y-axis grid and label]
- [-v|--vertical-label string] [-w|--width pixels]
- [--right-axis scale:shift] [--right-axis-label label]
- [--right-axis-format format]
- [-h|--height pixels] [-o|--logarithmic]
- [-u|--upper-limit value] [-z|--lazy]
- [-l|--lower-limit value] [-r|--rigid]
- [-g|--no-legend] [-d|--daemon <address>]
- [-F|--force-rules-legend]
- [-j|--only-graph]
- [-n|--font FONTTAG:size:font]
- [-m|--zoom factor]
- [-A|--alt-autoscale]
- [-M|--alt-autoscale-max]
- [-G|--graph-render-mode {normal,mono}]
- [-R|--font-render-mode {normal,light,mono}]
- [-B|--font-smoothing-threshold size]
- [-T|--tabwidth width]
- [-E|--slope-mode]
- [-P|--pango-markup]
- [-N|--no-gridfit]
- [-X|--units-exponent value]
- [-L|--units-length value]
- [-S|--step seconds]
- [-f|--imginfo printfstr]
- [-a|--imgformat PNG]
- [-c|--color COLORTAG#rrgbbb[aa]]
- [--border width]
- [-t|--title string]
- [-W|--watermark string]
- [-Z|--use-nan-for-all-missing-data]
- [DEF:vname=rrd:ds-name:CF]
- [CDEF:vname=rpn-expression]
- [VDEF:vdefname=rpn-expression]
- [PRINT:vdefname:format]
- [GPRINT:vdefname:format]
- [COMMENT:text]
- [SHIFT:vname:offset]
- [TEXTALIGN:{left|right|justified|center}]
- [TICK:vname#rrgbbb[aa][:[:fraction]][:legend]]
- [HRULE:value#rrgbbb[aa][:legend]]
- [VRULE:value#rrgbbb[aa][:legend]]
- [LINE[width]:vname[#rrgbbb[aa][[:legend]][:STACK]]]
- [AREA:vname[#rrgbbb[aa][[:legend]][:STACK]]]

## rrdgraph

■

```
#!/bin/bash
filename="/home/pi/temperatur1.rrd"
rrdtool graph temperatur1.png \
    --imgformat 'PNG' \
    --width 640 \
    --height 100 \
    --start -1hour \
    --end now \
    --vertical-label "Grad Celsius" \
    --alt-autoscale \
    --title Temperatur \
    DEF:templ=$filename:templ:AVERAGE \
    AREA:templ#CCCCFF: \
    LINE1:templ#0000FF:"Temperatur DS18B20" \
    GPRINT:templ:MIN:"Min\\:  %3.21f °C " \
    GPRINT:templ:MAX:"Max\\:  %3.21f °C " \
    GPRINT:templ:AVERAGE:"Avg\\: %3.21f °C " \
    GPRINT:templ:LAST:"Aktuell\\: %3.21f °C "
```



# Virtual Terminal

## screen

- Sitzung vorsetzen  
`screen -x`
- Neues Fenster  
`[Strg]+[a] [c]`
- Zum nächsten Fenster springen  
`[Strg]+[a] [n]`
- Zum vorherigen Fenster springen  
`[Strg]+[a] [p]`
- Sitzung trennen  
`[Strg]+[a] [d]`  
Anwendungen laufen im getrennten Zustand  
im Hintergrund weiter

## tmux

- Sitzung vorsetzen  
`tmux a`
- Neues Fenster  
`[Strg]+[b] [c]`
- Zum nächsten Fenster springen  
`[Strg]+[b] [n]`
- Zum vorherigen Fenster springen  
`[Strg]+[b] [p]`
- Sitzung trennen  
`[Strg]+[b] [d]`  
Anwendungen laufen im getrennten Zustand  
im Hintergrund weiter

Tipp: Tmux ist von Beiden das mächtigere Tool

## Plattenplatz berechnen

- `df -h <dir>`

Gibt Plattenbelegung und freien Platz des Dateisystems zurück, welches das Verzeichnis enthält.

Die `-h` -Option gibt es auch nur in GNU `df`

- Beispiel:

```
> df -h .
```

Dateisystem	Größe	Benutzt	Verf.	Verw%	Eingehängt auf
/dev/root	30G	5,1G	23G	19%	/
devtmpfs	434M	0	434M	0%	/dev
tmpfs	438M	45M	394M	11%	/run
/dev/mmcblk0p1	44M	23M	22M	52%	/boot

- `df -h`

Zeigt Platteninformationen über alle Dateisysteme im System. Nützlich, um bei Fehlern nach vollen Dateisystemen zu suchen.

## Der Befehl sort

- `sort <file>`  
Sortiert die Zeilen der Datei in Zeichenfolge und gibt sie aus
- `sort -r <file>`  
Das gleiche, aber in umgekehrter Reihenfolge
- `sort -ru <file>`  
u: unique. Das gleiche, gibt identische Zeilen aber nur einmal aus.
- `sort -n <file>`  
Numerische Sortierung, ansonsten ist es nach String

## Der Befehl tee

```
tee [-a] file
```

- Der `tee` -Befehl kann benutzt werden, um die Standardausgabe gleichzeitig auf den Bildschirm und in eine Datei zu senden
- `make | tee build.log`  
**Startet den `make` Befehl und speichert die Ausgabe in `build.log`**
- `make install | tee -a build.log`  
**Startet `make install` und fügt die Ausgabe an `build.log`**

## Der wget-Befehl

Anstatt Dateien mit Ihrem Browser abzurufen, kopieren Sie einfach die URL und rufen Sie die Datei mit `wget` ab!

### Eigenschaften von wget

- http und ftp-Unterstützung (auch mit SSL Verschlüsselung)
- Kann abgebrochene Übertragung fortsetzen
- Kann ganze Webseiten abrufen oder zumindest auf defekte Links testen
- Sehr nützlich in Skripten oder bei grafiklosen Systemen (System-Administration, eingebettete Systeme)
- Proxy-Unterstützung (`http_proxy` und `ftp_proxy` Umgebungsvariablen)
- Alternative: `curl`

## wget Beispiele

- `wget -c https://download.manjaro.org/xfce/22.0.3\`  
`/manjaro-xfce-22.0.3-230213-linux61.iso`

**Setzt einen abgebrochenen Abruf fort**

- `wget -m https://www.heise.de/`

**Spiegelt eine Seite**

- `wget -r -np http://www.xml.com/ldd/chapter/book/`

**Rekursiver Abruf eines Online-Buches für lokalen Zugriff.**

`-np`: "no-parent". Folgt nur Links im aktuellen Verzeichnis.

## /etc/rc.local

- Ist die Autostart Datei unter Linux
- Bash Skript das beim Starten ausgeführt wird
- Das Skript muss zu Ende kommen, ansonsten kann der Bootvorgang nicht abgeschlossen werden. D.h. Prozesse z.B. mit & in den Hintergrund abhängen.
- Der Return-Wert wird ausgewertet

## Aufgabe

- Bitte editiert die Datei so, dass beim nächsten Reboot das Temperatur Monitoring Skript wieder automatisch die Daten im Hintergrund aufzeichnet.

## Lösung Temperaturmonitoring in die Autostart mit aufnehmen

- Diese Zeilen in der `/etc/rc.local` einfügen
- `su pi -c 'tmux new -s "RRD Update Skripte" -d'`
- `su pi -c 'tmux set remain-on-exit on'`
- `su pi -c 'tmux new-window "/usr/local/bin/update_temp.sh"'`



## Webserver

Die drei gängigsten Webserver unter Linux sind Apache2, nginx und lighttpd  
Der ressourcensparende davon ist der Lighttpd, den wir verwenden werden.

- `sudo apt install lighttpd`

Der Speicherort der Webseite liegt unter:

`/var/www/html/`

Wie erzeugen dort eine index.html und rufen sie danach mit der Browser auf.

10.44.7.[101-116]

## Webserver CGI aktivieren

Da der Webseite Skripte ausführen darf,  
 Ist es notwendig es explizit zu erlauben.

- `sudo lighty-enable-mod cgi`

### Edit Webserver Konfiguration

- `sudo nano /etc/lighttpd/conf-available/10-cgi.conf`

- `sudo mkdir /var/www/html/cgi-bin`

- `sudo chown www-data:www-data /var/www/html/cgi-bin`

- `sudo service lighttpd restart`

```
# /usr/share/doc/lighttpd/cgi.txt

server.modules += ( "mod_cgi" )

$HTTP["url"] =~ "^/cgi-bin/" {
    cgi.assign = ( ".sh" => "/bin/bash" )
#    alias.url += ( "/cgi-bin/" => "/usr/lib/cgi-bin/" )
}

## Warning this represents a security risk, as it allow to e
## with a .pl/.py even outside of /usr/lib/cgi-bin.
#
#cgi.assign      = (
#    ".pl"      => "/usr/bin/perl",
#    ".py"      => "/usr/bin/python",
#)
```

## CGI Graph

```
#!/bin/bash
```

```
filename="/home/pi/temperatur1.rrd"
```

```
rrdtool graph - \  
    --imgformat 'PNG' \  
    --width 640 \  
    --height 100\  
    --start -1hour \  
    --end now \  
    --vertical-label "Grad Celsius" \  
    --alt-autoscale \  
    --title Temperatur \  
    DEF:temp1=$filename:temp1:AVERAGE \  
    AREA:temp1#CCCCFF: \  
    LINE1:temp1#0000FF:'Temperatur DS18B20' \  
    GPRINT:temp1:MIN:"Min\\: %3.2lf °C " \  
    GPRINT:temp1:MAX:"Max\\: %3.2lf °C " \  
    GPRINT:temp1:AVERAGE:"Avg\\: %3.2lf °C " \  
    GPRINT:temp1:LAST:"Aktuell\\: %3.2lf °C "
```

## Webseite erstellen mit Autorefresh

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Temperaturverlauf</title>
    <meta http-equiv="refresh" content="60">
  </head>
  <body>
    
  </body>
</html>
```

## Ende Teil eins – Linux BASH

- Im Netzwerkordner liegen diverse Linux Cheat Sheet

## Teil 2 - Python



# Python

- Schnell zu erlernende, moderne Programmiersprache
- Viele Standardfunktionen („all batteries included“)
- Bibliotheken für alle anderen Zwecke
- Freie Software mit aktiver Gemeinde
- Portabel, gibt es für fast jedes Betriebssystem
- Kommt in TIOBE auf Platz 1 (Platz 3 2021) \*
- Entwickelt von Guido van Rossum, CWI, Amsterdam

Feb 2024	Feb 2023	Change	Programming Language		Ratings	Change
1	1		 Python		15.16%	-0.32%
2	2		 C		10.97%	-4.41%
3	3		 C++		10.53%	-3.40%
4	4		 Java		8.88%	-4.33%
5	5		 C#		7.53%	+1.15%
6	7	▲	 JavaScript		3.17%	+0.64%
7	8	▲	 SQL		1.82%	-0.30%
8	11	▲	 Go		1.73%	+0.61%
9	6	▼	 Visual Basic		1.52%	-2.62%
10	10		 PHP		1.51%	+0.21%
11	24	▲	 Fortran		1.40%	+0.82%

\* Stand Februar 2024 - <https://www.tiobe.com/tiobe-index/>

## Informationen zu Python

- Aktuelle Versionen 3.12.2 bzw. 3.11.2 (3.11.7)
- 2.x ist noch weiter verbreitet wird aber nicht mehr supported (2.7.18)
- Diese Vorlesung behandelt daher nur 3.x

## Hilfe zu Python

- Offizielle Homepage  
<http://www.python.org>
- Ansonsten wie immer, Suchmaschine ist dein Freund



## Python starten

```
root@raspi:/# python3
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>> help(print)
>>> exit()
```

- >>> markiert Eingaben
- print: Ausgabe auf das Terminal
- help(): interaktive Hilfe, wird mit „q“ beendet
- Statt exit() reicht auch Control-d
- oder ipython mit Tab-Ergänzung, History usw.

# Aufgabe

- Schreibt in Python3 Hello World

## Lösung

```
#!/usr/bin/python3
```

```
# unsere erste Python-Anweisung  
print("Hello World")
```

- Mit python helloworld.py starten
- Oder ausführbar machen (chmod a+x helloworld.py)
- Umlaute vermeiden oder Encoding-Cookie einfügen  
Bei Python 3 ist utf-8 default
- „#!“ funktioniert genauso wie beim Shell-Skript
- Zeilen, die mit „#“ starten, sind Kommentare

## Datentyp: Zahlen

```
>>> print(42)
42
>>> print(42/3)
14.0
>>> print(42/12)
3.5
>>> print(42/12,-3/2)
3.5 -1.5
```

- Achtung Division mit Python 2 liefert nur ganzzahligen Rest
- print gibt mit Komma auch mehrere Werte aus

## Datentyp: (Fließ-)kommazahlen

```
>>> print(12345.000)
```

```
12345.0
```

```
>>> print(6.023e23, 13.8E-24)
```

```
6.023e+23 1.38e-23
```

```
>>> print(3.0/2)
```

```
1.5
```

- Reelle Zahlen der Form  $6,023 \cdot 10^{23}$
- $1.38e-23$  steht z. B. für  $1,38 \times 10^{-23}$
- Achtung: englische Schreibweise, Punkt statt Komma
- Keine Tausenderpunkte (oder -kommas)
- Endliche binäre Genauigkeit von **Mantisse** und **Exponent**
- 12345 ungleich 12345.0 (z. B. bei der Ausgabe)

## Datentyp: Zeichenketten

```
>>> print("Hello World")
Hello World
>>> print('Hello World')
Hello World
>>> print("""Hello
... World""")
Hello
World
```

- zwischen einfachen (') oder doppelten (") Anführungszeichen
- Über mehrere Zeilen mit dreifachen Anführungszeichen
- Zeichenketten sind keine Zahlen! "1"  $\neq$  1
- `int(string)` konvertiert Zeichenkette in Ganzzahl
- entsprechend `float(string)` für Fließkomma

## Variablen

```
>>> var1 = 2
>>> var2 = 4
>>> print(var1,var2)
2 4
>>> var1 = var1 * var2
>>> var2 = 3
>>> print(var1,var2)
8 3
```

- Werte können mit Namen belegt werden und verändert
- keine mathematischen Variablen, sondern Speicherplätze
- Daher ist `var1 = var1 * var2` kein Unsinn, sondern multipliziert `var1` mit `var2`
- Die nachträgliche Änderung von `var2` ändert nicht `var1`, das Ergebnis der vorherigen Rechnung!

## Variablen 2

- Variablennamen bestehen aus Buchstaben, Ziffern oder „\_“ (Unterstrich), am Anfang keine Ziffer
- Groß-/Kleinschreibung ist relevant: Hase  $\neq$  hase
- **Richtig:** i, some\_value, SomeValue, v123, \_hidden, \_1
- **Falsch:** 1\_value, some value, some-value
- Am besten sprechende Bezeichner, also „factorial“, „n“ statt „var1“, „var2“



## Arithmetische Ausdrücke

+	Addition, bei Strings aneinanderfügen, z.B. $1 + 2 \rightarrow 3$ , $"a" + "b" \rightarrow "ab"$
-	Subtraktion, z.B. $1 - 2 \rightarrow -1$
*	Multiplikation, Strings vervielfältigen, z.B. $2 * 3 = 6$ , $"ab" * 2 \rightarrow "abab"$
/	Division, z.B. $3 / 2 \rightarrow 1.5$ (Python 2 nur ganzzahlig)
%	Rest bei Division, z.B. $5 \% 2 \rightarrow 1$
**	Exponent, z.B. $3^{**}2 \rightarrow 9$ , $0.1^{**}3 \rightarrow 0.001$

- Mathematische Präzedenz (Exponent vor Punkt vor Strich),  
 z. B.  $2^{**}3 * 3 + 5 \rightarrow 2^3 \cdot 3 + 5 = 29$
- Präzedenz kann durch runde Klammern geändert werden:  
 $2^{**}(3 * (3 + 5)) \rightarrow 2^{3 \cdot 8} = 16.777.216$

## Aufgabe

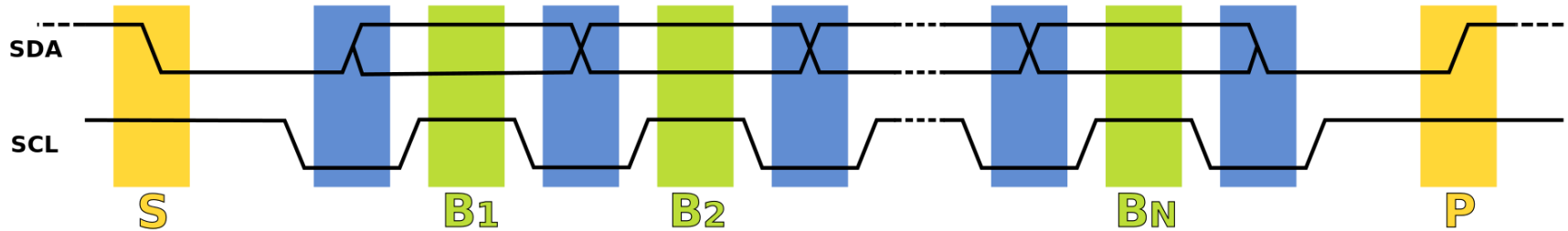
- Portieren sie das Bash Skript, das den RRD Graph erzeugt nach Python.

Voraussetzung Installation folgender Pakete:

- `sudo apt install librrd-dev`
- `sudo apt install python3-rrdtool (or pip3 install rrdtool)`

## i<sup>2</sup>C (Inter-Integrated Circuit)

- Erfunden 1982 von Philips Semiconductors (heute NXP Semiconductors)
- Serieller Datenbus
- Kommunikation zwischen Controller und Peripherie-ICs
- Patent ist 2006 abgelaufen, inzwischen über 50 Hersteller mit mehr als 1000 verschiedene ICs
- Atmel nennt in TWI (Two-Wire-Interface)
- Master-Slave-Bus der auch Multimaster fähig ist
- Adressierung ist 7 Bit (16 Adressen reserviert für Sonderzwecke)
- 112 Knoten in einem Bus
- 2 Signalleitungen
  - SCL = Serial Clock
  - SDA = Serial Data



Zwischen dem *Start*-Signal (S) und dem *Stopp*-Signal (P) werden die Datenbits  $B_1$  bis  $B_N$  übertragen.

## i2c Bus Bash Commando

```
sudo apt install i2c-tools
```

Um den Bus abzuscannen ob I2C Geräte angeschlossen sind und wie deren Slave Adresse lautet

- `i2cdetect -y 1`
  
- Zum Lesen der Daten  
`i2cdump` oder `i2cget`
  
- Zum Schreiben Daten an einen Slave  
`i2cset`

## BMP280

- Package
  - 2.5mm x 2.5mm x 0.93mm metal lid LGA
- Digital interface
  - I<sup>2</sup>C (up to 3.4 MHz)
  - SPI (3 and 4 wire, up to 10 MHz)
- Supply voltage
  - Supply voltage range: 1.71V to 3.6V
- Current consumption
  - 3.6µA @ 1Hz humidity, pressure and temperature
- Operating range
  - -40...+85 °C
  - 300...1100 hPa

## Aufgabe

- Bestimmt die Temperatur und Luftdruck

# Lösung 1

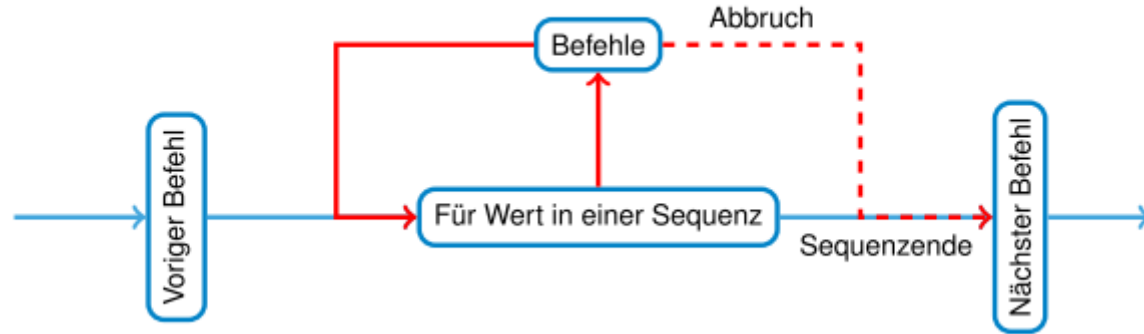
- `#!/usr/bin/env python3`
- `import smbus`
- `import time`
- `# number of I2C bus`
- `BUS = 1`
- `# BMP280 address, 0x76 or 0x77`
- `BMP280ADDR = 0x77`
- `# altitude 500 m`
- `ALTITUDE = 120`
- `# get I2C bus`
- `bus = smbus.SMBus(BUS)`
- `# temperature calibration coeff. array`
- `T = [0, 0, 0];`
- `# pressure calibration coeff. array`
- `P = [0, 0, 0, 0, 0, 0, 0, 0];`
- `# read calibration data from 0x88, 24 bytes`
- `data = bus.read_i2c_block_data(BMP280ADDR, 0x88, 24)`
- `# temp coefficients`
- `T[0] = data[1] * 256 + data[0]`
- `T[1] = data[3] * 256 + data[2]`
- `if T[1] > 32767:`
- `T[1] -= 65536`
- `T[2] = data[5] * 256 + data[4]`
- `if T[2] > 32767:`
- `T[2] -= 65536`
- `# pressure coefficients`
- `P[0] = data[7] * 256 + data[6];`
- `for i in range(0, 8):`
- `P[i+1] = data[2*i+9]*256 + data[2*i+8];`
- `if P[i+1] > 32767:`
- `P[i+1] -= 65536`
- `# select control measurement register, 0xF4`
- `# 0x27: pressure/temperature oversampling rate = 1, normal mode`
- `bus.write_byte_data(BMP280ADDR, 0xF4, 0x27)`
- `# select configuration register, 0xF5`
- `# 0xA0: standby time = 1000 ms`
- `bus.write_byte_data(BMP280ADDR, 0xF5, 0xA0)`
- `time.sleep(1.0)`



## Lösung 2

- # read data from 0xF7, 8 bytes
- data = bus.read\_i2c\_block\_data(BMP280ADDR, 0xF7, 8)
- # convert pressure and temperature data to 19 bits
- adc\_p = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
- adc\_t = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
- # temperature offset calculations
- temp1 = ((adc\_t)/16384.0 - (T[0])/1024.0)\*(T[1]);
- temp3 = (adc\_t)/131072.0 - (T[0])/8192.0;
- temp2 = temp3\*temp3\*(T[2]);
- temperature = (temp1 + temp2)/5120.0
- # pressure offset calculations
- press1 = (temp1 + temp2)/2.0 - 64000.0
- press2 = press1\*press1\*(P[5])/32768.0
- press2 = press2 + press1\*(P[4])\*2.0
- press2 = press2/4.0 + (P[3])\*65536.0
- press1 = ((P[2])\*press1\*press1/524288.0 + (P[1])\*press1)/524288.0
- press1 = (1.0 + press1/32768.0)\*(P[0])
- press3 = 1048576.0 - (adc\_p)
- if press1 != 0:
  - press3 = (press3 - press2/4096.0)\*6250.0/press1
  - press1 = press3\*press3\*(P[8])/2147483648.0
  - press2 = press3\*(P[7])/32768.0
  - pressure = (press3 + (press1 + press2 + (P[6]))/16.0)/100
- else:
  - pressure = 0
- # pressure relative to sea level
- pressure\_nn = pressure/pow(1 - ALTITUDE/44330.0, 5.255)
- # output data to screen
- print("Temperature: %.2f C" %temperature)
- print("Pressure: %.2f hPa " %pressure)
- print("Pressure NN: %.2f hPa " %pressure\_nn)

# For-Schleifen



- Wiederholen eines Blocks von Befehlen
- *Schleifenvariable* nimmt dabei verschiedene Werte aus einer *Sequenz* (Liste) an
- Die abzuarbeitende Sequenz bleibt fest
- Kann bei Bedarf abgebrochen werden (Ziel erreicht, Fehler, . . . )

## For- Schleifen

for Variable in Sequenz:

    Anweisung\_1

    Anweisung\_2

    ...

    Anweisung\_n

else:

    Else-Anweisung\_1

    Else-Anweisung\_2

    ...

    Else-Anweisung\_m

```
>>> languages = ["C", "C++", "Perl",  
"Python"]
```

```
>>> for language in languages:
```

```
...     print(language)
```

```
...
```

```
C
```

```
C++
```

```
Perl
```

```
Python
```

## Range

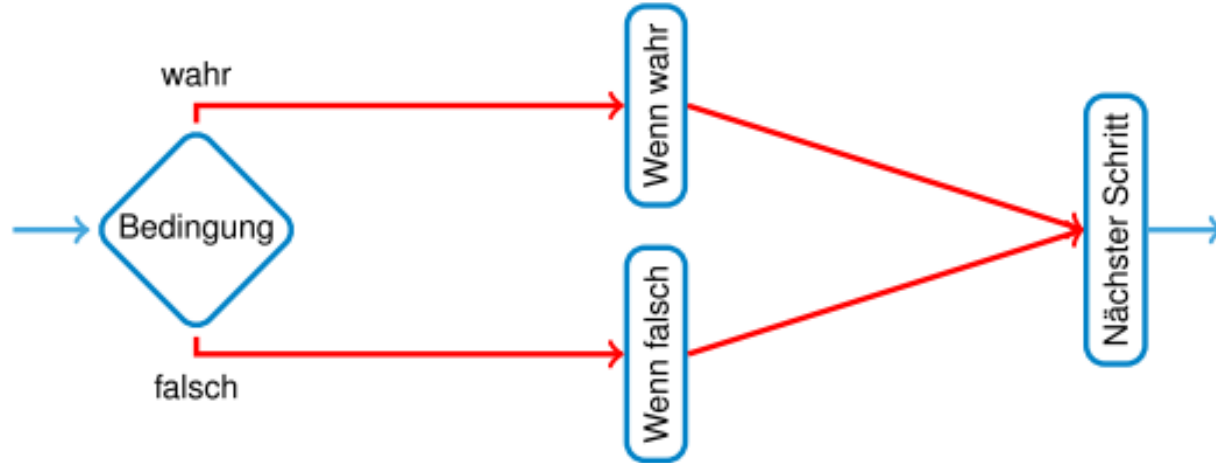
- Für Zählerschleifen verwenden man am einfachsten die range-Funktion

- `range(start, stop[, step])`

<code>range(10)</code>	<code>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]</code>
<code>range(3,12)</code>	<code>[3, 4, 5, 6, 7, 8, 9, 10, 11]</code>
<code>range(5,50,5)</code>	<code>[5, 10, 15, 20, 25, 30, 35, 40, 45]</code>
<code>range(10,0,-1)</code>	<code>[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]</code>

- Mit `list(range(10))` kann man am einfachsten die Werte ausgeben

## Bedingte Ausführung



- Das Programm kann auf Werte von Variablen verschieden reagieren
- Wird als Verzweigung bezeichnet
- Auch mehr Äste möglich (z.B.  $< 0$ ,  $= 0$ ,  $> 0$ )

## IF: Bedingte Ausführung in Python

```
a = 1
if a < 5:
    print("a ist kleiner als 5")
elif a > 5:
    print("a ist groesser als 5")
else:
    print("a ist 5")
```

- if-elif-else führt den Block nach der ersten erfüllten
- Bedingung (logischer Wert True) aus
- Trifft keine Bedingung zu, wird der else-Block ausgeführt
- elif oder else sind optional

## Logische Ausdrücke

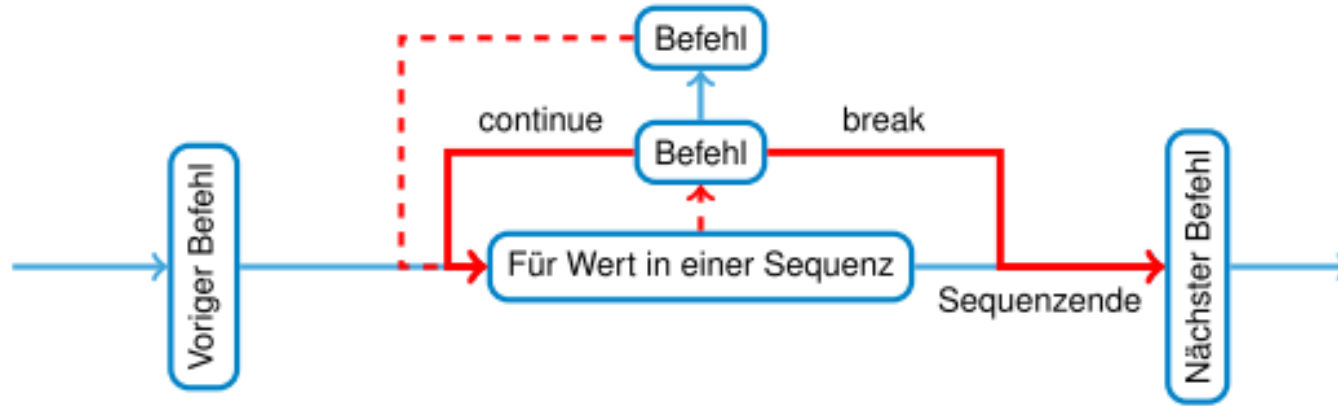
==, !=	Test auf (Un-)Gleichheit, z.B. $2 == 2 \rightarrow \text{True}$ , $1 == 1.0 \rightarrow \text{True}$ , $2 == 1 \rightarrow \text{False}$
<, >, <=, >=	Vergleich, z.B. $2 > 1 \rightarrow \text{True}$ , $1 <= -1 \rightarrow \text{False}$
or, and	Logische Verknüpfungen „oder“ bzw. „und“
not	Logische Negation, z.B. $\text{not False} == \text{True}$

- Wahrheitswerte True („wahr“) oder False („falsch“)
- Verknüpfungen wie in der klassischen Aussagenlogik
- Präzedenz: logische Verknüpfungen vor Vergleichen

$3 > 2 \text{ and } 5 < 7 \rightarrow \text{True}$

$1 < 1 \text{ or } 2 >= 3 \rightarrow \text{False}$

## For-Schleife: else, break und continue



- **break:** Die Schleife wird sofort verlassen
- **continue:** Der aktuelle Schleifendurchgang wird unterbrochen, aber mit dem nächsten Element fortgesetzt
- **else:** Wird nur aufgerufen, wenn alle Element durchgelaufen sind (kein break Aufruf)



## For-Schleife: Beispiele

```
for zahl in range(10):
    if zahl == 3 :
        print("Oh nein, eine drei!")
        break
    print(zahl)
else:
    print("Die Zahl drei ist nicht
vorgekommen")
print("Das Programm läuft weiter.")
```

0

1

2

Oh nein, eine drei!

Das Programm läuft weiter.

```
▪ for zahl in range(10):
▪     if zahl == 3 :
▪         print("Oh nein, eine drei!")
▪         continue
▪     print(zahl)
▪ else:
▪     print("Kein break gefunden.")
▪ print("Das Programm läuft weiter.")
```

▪ 0

▪ 1

▪ 2

▪ Oh nein, eine drei!

▪ 4

▪ 5

▪ 6

▪ 7

▪ 8

▪ 9

▪ Kein break gefunden.

▪ Das Programm läuft weiter.

## Blöcke und Einrückung

```
>>> b = 0
```

```
>>> for a in range(1, 4):
```

```
... b = b + a
```

```
... print(b)
```

```
4
```

```
6
```

```
9
```

```
>>> b = 0
```

```
>>> for a in range(1, 3): b = b + a
```

```
... print(b)
```

```
9
```

- Alle gleich eingerückten Befehle gehören zum Block
- Einzeilige Blöcke können auch direkt hinter den Doppelpunkt stehen
- Einrücken durch Leerzeichen oder Tabulatoren

## Blöcke und Einrückung

- ein Block kann nicht leer sein, aber der Befehl `pass` tut nichts:

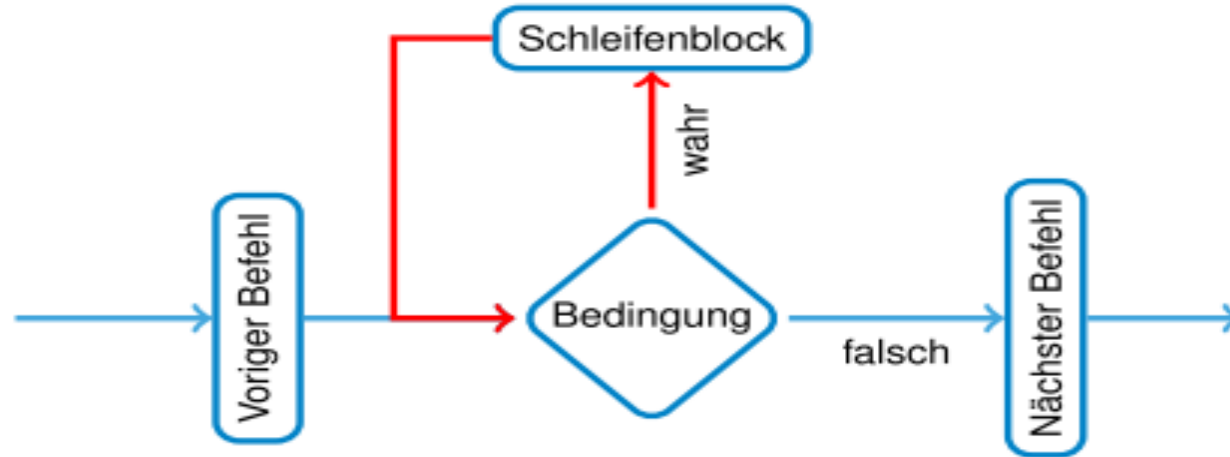
```
if a < 5:  
    pass  
  
else:  
    print("a ist groesser gleich 5")
```

- Häufige Fehlermeldung:

`IndentationError: unexpected indent`

- Bitte kontrolliert eure Einrückungen.
- Falsche Einrückung führt im allgemeinen zu Programmfehlern!

# While-Schleifen



- Wiederholte Ausführung ähnlich wie for-Schleifen
- Keine Schleifenvariable, sondern Schleifenbedingung
- Ist die Bedingung immer erfüllt, kommt es zur Endlosschleife

Solange  $a > 0$ , ziehe eins von  $a$  ab

Solange noch Zeilen in der Datei sind, lese eine Zeile

## While-Schleifen in Python

```
>>> a = 1
```

```
>>> while a < 5:
```

```
...     a = a + 1
```

```
>>> print(a)
```

5

- Führt den Block solange aus, wie die Bedingung wahr ist
- Block wird nicht ausgeführt, wenn Bedingung sofort verletzt ist:

```
>>> a = 6
```

```
>>> while a < 5:
```

```
...     a = a + 1
```

```
...     print("Erhöhe a um eins")
```

```
>>> print(a)
```

6

## While: Beispiel

```
import random
n = 20
to_be_guessed = random.randint(1,n)

guess = 0
while guess != to_be_guessed:
    guess = int(input("Neuer Versuch: "))
    if guess > 0:
        if guess > to_be_guessed:
            print("Zu gross")
        elif guess < to_be_guessed:
            print("Zu klein")
    else:
        print("Schade, dass du aufgibst!")
        break
else:
    print("Gratuliere, das war's")
```

Beispiel liegt auch auf dem Webserver: <http://10.44.7.117/while.py>

## Formatierte Ausgabe: Alte Methode %

- Alte Methode im Stil von printf und sprintf
- Funktioniert noch, die Frage ist aber wie lange
- Daher wir hier nicht genauer beschrieben

```
>>> print("Nummer: %5d, Preis: %8.2f" % (123, 19.589))  
Nummer:    123, Preis:    19.59
```

## Formatierte Ausgabe: .format

- Der pythonische Weg
- String.format ersetzt {}-Platzhalter
- erster Parameter ersetzt {0}, zweiter {1} usw.
- Formatierungsangaben hinter „:“, u. a.
  - <, >, ^: links-, rechtsbündig oder zentriert
  - +: Vorzeichen immer ausgeben
  - 0: führende Nullen bei Zahlen
  - gefolgt von Breite.Genauigkeit
  - e, f, g: verschiedene Fließkommaformate
  - x, b: hexadezimal und binär ausgeben

```
>>> print("{}^2 + {}^2 = {:05}^2".format(3, 4, 5))
```

```
3^2 + 4^2 = 00005^2
```

```
>>> print("Strings {1} {0:>10}".format("Welt", "Hallo"))
```

```
Strings Hallo          Welt
```

```
>>> print("Fliess {:e} {:+8.4f} {:g}".format(3.14,3.14,3.14))
```

```
Fliess 3.140000e+00  +3.1400 3.14
```



## Aufgabe

- Erzeuge eine weitere RRD Datenbank für den BMP280 Sensor, die die Temperatur und Luftdruck alle 5 Sekunden speichern kann.
- Die Werte sollten von der Average Auflösung:
  - 1 Tag                5s
  - 1 Woche            60s
  - 1 Monat            10min
  - 1 Jahr              1TagMin/Max nach belieben.
- Verwenden sie, das zuvor erstellte Skript um die Daten in die Datenbank zu füttern.

Voraussetzung Installation folgender Pakete (sollten aber bereits installiert sein):

- `sudo apt install librrd-dev`
- `sudo apt install python3-rrdtool`

## Aufgabe

- Generieren Sie mit dem Webserver zwei Bilder
- Bild 1 zeigt die beiden Temperaturkurven
- Bild 2 zeigt den Luftdruck.

## Parameter einlesen

```
import sys
# get integer c from the command line
try:
    c = int(sys.argv[1])
except:
    sys.stderr.write("usage: {} <c>\n".format(sys.argv[0]))
    exit(-1)
print c
```

- Bisher ist c fest im Programm  $\Rightarrow$
- wenn wir c ändern wollen, müssen wir das Programm ändern
- Besser von der Kommandozeile lesen!
- So können wir das Programm direkt vom Terminal benutzen
- Wir brauchen keinen Editor, wenn es mal tut

## Parameter einlesen

```
import sys
# get integer c from the command line
try:
    c = int(sys.argv[1])
except:
    sys.stderr.write("usage: {} <c>\n".format(sys.argv[0]))
    exit(-1)
print c
```

- import sys lädt das sys-Modul, dazu später mehr
- sys.argv[i] gibt dann den i-ten Parameter des Programms
- sys.argv[0] ist der Name des Skripts
- int(string) konvertiert Zeichenkette in Ganzzahl
- Der except-Block wird nur ausgeführt, wenn es beim Lesen von
- c einen Fehler gab

## Beispiel: Sortieren

- Gegeben: Liste  $A = a_0, \dots, a_N$
- Gesucht: Liste  $A$  aufsteigen
- Datentyp ist egal, so lange  $\leq$  definiert ist
- In Python ganz einfach:
  - $A.sort() \Rightarrow A$  wird umsortiert
  - $B = \text{sorted}(A) \Rightarrow A$  bleibt gleich,  $B$  ist die sortierte Liste

```
>>> A = [2,1,3,5,4]
>>> print(sorted(A), A)
[1, 2, 3, 4, 5] [2, 1, 3, 5, 4]
>>> A.sort()
>>> print(A)
[1, 2, 3, 4, 5]
```

## Listen in Python

```
>>> kaufen = [ "Muesli", "Milch", "Obst" ]
>>> kaufen[1] = "Sahne" # "Milch" durch Sahne ersetzen
>>> print(kaufen)
['Muesli', 'Sahne', 'Obst']
>>> print(kaufen[0])
Muesli
>>> print(kaufen[-1])
Obst
>>> print("Saft" in kaufen)
False
```

- Komma-getrennt in eckigen Klammern
- Können Daten verschiedenen Typs enthalten
- Liste[i] bezeichnet das i-te Listenelement (bei 0 startend)
- Negative Indizes starten vom Ende
- X in Liste überprüft, ob Liste ein Element mit Wert X enthält

## Elemente zu Listen hinzufügen

```
>>> kaufen = [ "Muesli", "Milch", "Obst" ]
>>> kaufen.append("Brot")
>>> print(kaufen)
['Muesli', 'Milch', 'Obst', 'Brot']
>>> kaufen.insert(1, "Saft")
>>> print(kaufen)
['Muesli', 'Saft', 'Milch', 'Obst', 'Brot']
>>> kaufen = kaufen + [ "Milch", "Mehl" ]
>>> print(kaufen)
['Muesli', 'Saft', 'Milch', 'Obst', 'Brot', 'Milch', 'Mehl']
```

- `Liste.append(x)` hängt `x` am Ende der Liste an
- `Liste.insert(i, x)` fügt `x` an Position `i` der Liste an
- Listen können durch `+` aneinandergefügt werden

## Elemente löschen

```
>>> kaufen = ['Muesli', 'Saft', 'Milch', 'Obst', 'Brot', 'Milch']
>>> kaufen.remove("Milch")
>>> print(kaufen)
['Muesli', 'Saft', 'Obst', 'Brot', 'Milch']
>>> kaufen.remove("Milch")
>>> print(kaufen)
['Muesli', 'Saft', 'Obst', 'Brot']
>>> kaufen.remove("Mehl")
ValueError: list.remove(x): x not in list
>>> del kaufen[-1]
>>> print(kaufen)
['Muesli', 'Saft', 'Obst']
```

- `Liste.remove(x)` entfernt das erste Element mit dem Wert `x` aus der Liste
- Fehler, wenn es kein solches gibt (daher mit `in` testen `if "Mehl" in kaufen: kaufen.remove("Mehl")`)
- `del liste[i]` löscht das Element mit Index `i` aus der Liste



## Unterlisten

```
>>> kaufen = [ "Muesli", "Sahne", "Obst", "Oel", "Mehl" ]
>>> print(kaufen[3:4])
['Oel']
>>> for l in kaufen[1:3]:
...     print(l)
Sahne
Obst
>>> print(len(kaufen[:4]))
4
```

---

- `[i:j+1]` ist die Unterliste vom i-ten bis zum j-ten Element
- Leere Grenzen entsprechen Anfang bzw. Ende,
- also stets `liste == liste[:]` == `liste[0:]`
- for-Schleife iteriert über alle Elemente
- negative Indizes starten vom Ende
- `len()` berechnet die Listenlänge

## Vorsicht: Flache Kopien!

```
>>> kaufen = [ "Muesli", "Milch", "Obst" ]
```

### Flache Kopie:

```
>>> merken = kaufen  
>>> del kaufen[-1]  
>>> print(merken)  
['Muesli', 'Milch']
```

### Subliste, echte Kopie:

```
>>> merken = kaufen[:]  
>>> del kaufen[-1]  
>>> print(merken)  
['Muesli', 'Milch', 'Obst']
```

**„=“ macht in Python flache Kopien von Listen!**

- Flache Kopien (shallow copies) verweisen auf dieselben Daten
- Änderungen an einer flachen Kopie betreffen auch das Original
- Sublisten sind echte Kopien (deep copies, weil alles kopiert wird)
- Nur `kaufen[:]` ist eine echte Kopie von `kaufen`

## Tupel: unveränderbare Listen

```
>>> kaufen = "Muesli", "Kaese", "Milch"
>>> for f in kaufen: print(f)
Muesli
Kaese
Milch
>>> kaufen[1] = "Camembert"
TypeError: 'tuple' object does not support item assignment
>>> print (kaufen + ("Kaese", "Milch"))
('Muesli', 'Kaese', 'Milch', 'Kaese', 'Milch')
```

---

- Komma-getrennt in runden Klammern
- Solange eindeutig, können die Klammern weggelassen werden
- Können nicht verändert werden
- Ansonsten wie Listen einsetzbar, aber schneller
- Zeichenketten sind Tupel von Zeichen

## Austauschen (Swappen) von Werten mit Tupeln

- `>>> A=1`
- `>>> B=2`
- `>>> A, B = B, A`
  
- `>>> print(A, B)`
- `2 1`

So hingegen nicht:

```
>>> A=1
>>> B=2
>>> A = B
>>> B = A
>>> print(A, B)
2 2
```

- Listen und Tupel können links vom Gleichheitszeichen stehen
- Elemente werden der Reihe nach zugeordnet
- `A,B = B,A` tauscht also die Werte zweier Variablen aus (Tupelzuweisung!)

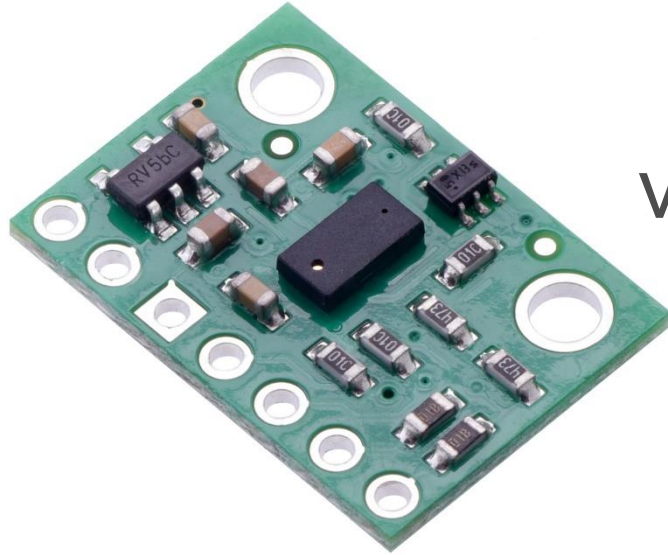
## Listen aus Listen erzeugen

```
■ >>> print([a**2 for a in [0,1,2,3,4]])  
■ [0, 1, 4, 9, 16]  
■ >>> print(sum([a**2 for a in range(5)]))  
■ 30  
■ >>> print([a for a in range(10) if a % 2 == 1])  
■ [1, 3, 5, 7, 9]  
■ >>> print([(a,b) for a in range(3) for b in range(2)])  
■ [(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

---

- Listen können in neue Listen abgebildet werden
- Syntax: [Ausdruck for Variable in Liste if Bedingung]
- Ausdruck: beliebige Formel, die meist von variable abhängt
- Variable,Liste: wie in einer for-Schleife
- Bedingung: welche Werte für variable zulässig sind
- Mehrere fors können aufeinander folgen (rechteckiges Schema)

## VL53L0X



versus

## HC-SR04



- IR Laser 940nm TOF (Time-of-Flight)
- Reichweite 2-125cm
- Abstrahlwinkel: 25°

- Ultraschall
- Reichweite 2- 400cm
- Abstrahlwinkel: 30°

## Vergleich VL53L0X versus HC-SR04

### ■ Pro

- Kleine Bauform
- Bussystem (einfach auszulesen, mehrere Sensoren an einer Datenleitung)
- Geringerer Abstrahlwinkel (Für einen LASER aber gigantisch)
- Funktioniert auch wenn das Objekt im spitzen Winkel steht.

### ■ Contra

- Preis 3,30€ gegenüber HC-SR04 1,-€ \*
- Reichweite, realistisch 80cm gegenüber 150cm

\* Preis Ebay Stand 05/2019

## Aufgabe VL53L0X

- Bitte misst mit Hilfe des VL53L0X den Abstand zwischen Tischplatte (RPI Versuchsplatine) und der Ablage darüber. Bitte verwendet dazu ein Python3 Programm.
- Hilfsmittel alles erlaubt!
- Als Erinnerung, zum Auslesen der i2c Adresse:  
`i2cdetect -y 1`

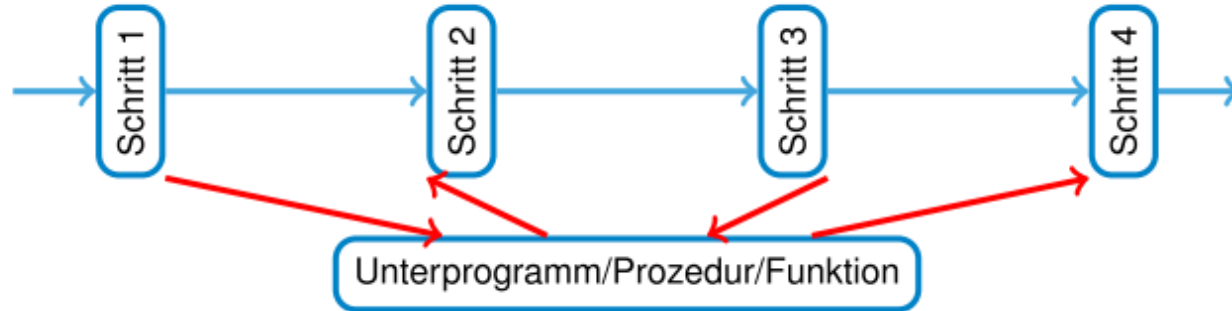


## Lösung

```
git clone https://github.com/johnbryanmoore/VL53L0X_rasp_python.git VL53L0X
cd VL53L0X
make
cd python
./VL53L0X_example.py
```

für VL53L0X\_example\_livegraph.py wird noch die plot Library benötigt  
sudo apt install python3-matplotlib

# Funktionen



- Funktionen (Unterprogramme, Prozeduren) unterbrechen die aktuelle Befehlskette und fahren an anderer Stelle fort
- Kehren an ihrem Ende wieder zur ursprünglichen Kette zurück
- Funktionen können selber wieder Funktionen aufrufen
- Vermeiden Code-Duplikation
  - kürzerer Code
  - besser wartbar, Fehler müssen nur einmal verbessert werden
- Sprechende Namen dokumentieren, was der Code Teil tun soll

## Funktionen in Python

```
>>> def printPi():  
...     print("pi ist ungefaehr 3.14159")  
>>> printPi()  
pi ist ungefaehr 3.14159
```

```
>>> def printMax(a, b):  
...     if a > b: print(a)  
...     else: print(b)  
>>> printMax(3, 2)  
3
```

- 
- Eine Funktion kann beliebig viele Argumente haben
  - Argumente verhalten sich wie Variablen
  - Beim Aufruf bekommen die Argumentvariablen Werte in der Aufrufreihenfolge
  - Der Funktionskörper ist wieder ein Block

## Lokale Variablen

```
>>> def max(a, b):  
...     if a > b: maxVal=a  
...     else: maxVal=b  
...     print(maxVal)  
>>> max(3,2)  
3  
>>> print(maxVal)  
NameError: name 'maxVal' is not defined
```

```
>>> faktor=2  
>>> def strecken(a):  
...     print(faktor*a)  
>>> strecken(1.5)  
3.0
```

- Neue Variablen innerhalb einer Funktion sind lokal
- Existieren nur während der Funktionsausführung
- Globale Variablen können nur gelesen werden

## Vorgabewerte und Argumente benennen

```
>>> def lj(r, epsilon = 1.0, sigma = 1.0):  
...     return 4*epsilon*( (sigma/r)**6 - (sigma/r)**12 )  
>>> print(lj(2**(1./6.)))  
1.0  
>>> print(lj(2**(1./6.), 1, 1))  
1.0
```

- 
- Argumentvariablen können mit Standardwerten vorbelegt werden
  - diese müssen dann beim Aufruf nicht angegeben werden
  - beim Aufruf können die Argumente auch explizit belegt werden, dann ist die Reihenfolge egal
- 

```
>>> print(lj(r = 1.0, sigma = 0.5))  
0.0615234375  
>>> print(lj(epsilon=1.0, sigma = 1.0, r = 2.0))  
0.0615234375
```

## return: eine Funktion beenden

```
>>> def max(a, b):  
...     if a > b: return a  
...     else: return b  
>>> print(max(3, 2))  
3
```

- 
- **return** beendet die Funktion sofort (vgl. **break**)
  - eine Funktion kann einen Wert zurückliefern
  - der Wert wird bei **return** spezifiziert
-

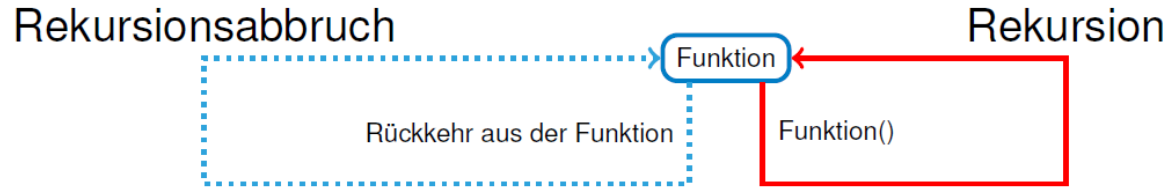
## Dokumentation von Funktionen

```
def max(a, b):  
    "Gibt das Maximum von a und b aus."  
    if a > b: print a  
    else: print b  
def min(a, b):  
    ""  
    Gibt das Minimum von a und b aus. Funktioniert  
    ansonsten genau wie die Funktion max.  
    ""  
    if a < b: print a  
    else: print b
```

- Dokumentation optionale Zeichenkette vor dem Funktionskörper
- Wird bei `help(funktion)` ausgegeben

# Rekursion

Eine Funktion, die sich selber aufruft, heißt **rekursiv**

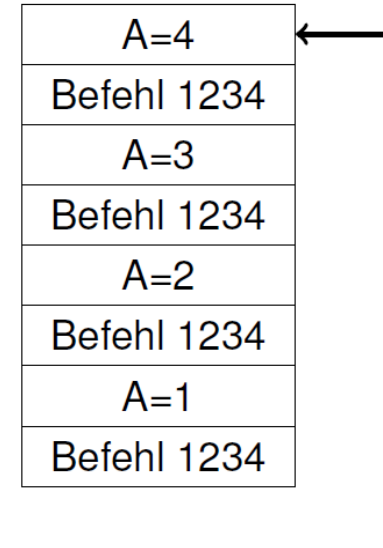


- Viele Algorithmen lassen sich elegant als Rekursion formulieren
- Ablauf meist nicht einfach zu verstehen
- Ob eine Rekursion endet, ist nicht immer offensichtlich
- Benötigt stets Abbruchkriterium!



## Funktionsaufrufe und Stack

- Wieso werden Funktionswerte bei Rekursion nicht überschrieben?
- Speicherung bei Funktionsaufruf auf dem **Stack**
- Eine Liste, bei der nur am Ende hinzugefügt (**push**) oder entfernt (**pop**) werden kann
- Jede Rekursion kann mit Hilfe eines Stacks in Schleifen übersetzt werden
- Rücksprungadresse auf dem Stack speichern
- So wird Rekursion in Computern tatsächlich umgesetzt



## Wörterbücher (dicts)

```
>>> de_en = { "Milch": "milk", "Mehl": "flour" }  
>>> de_en["Eier"]="eggs"  
>>> print(de_en["Milch"])  
milk  
>>> if "Mehl" in de_en: print("I can translate \"Mehl\"")  
I can translate "Mehl"
```

- 
- Komma-getrennte Paare von Schlüsseln (Keys) und Werten in geschweiften Klammern
  - Die Werte sind zu den Schlüsseln **assoziiert**
  - Vergleiche Wörterbuch: Deutsch  $\Rightarrow$  Englisch
  - Mit **in** kann nach Schlüsseln gesucht werden
  - Gut für unstrukturierte Daten

## Wörterbücher (dicts)

```
>>> for de in de_en: print(de, "=>", de_en[de])
Eier => eggs
Mehl => flour
Milch => milk
>>> de_en["Mehl"] = "wheat flour"
>>> for de, en in de_en.items(): print(de, "=>", en)
Eier => eggs
Mehl => wheat flour
Milch => milk
```

- 
- Werte sind änderbar (siehe auch Zählprogramm)
  - Indizierung über die Keys, nicht Listenindex o.ä.
  - **for** iteriert auch über die Schlüssel
  - Oder mit **items** über Schlüssel-Wert-Tupel (Python2 = iteritems)

## Stringmethoden

- Zeichenkette in Zeichenkette suchen

"Hallo Welt".**find**("Welt")  $\Rightarrow$  6

"Hallo Welt".**find**("Mond")  $\Rightarrow$  -1

- Zeichenkette in Zeichenkette ersetzen

" abcdabcabe ".**replace**("abc", "123")  $\Rightarrow$  '123d123abe'

- Groß-/Kleinschreibung ändern

"hallo".**capitalize**()  $\Rightarrow$  'Hallo'

"Hallo Welt".**upper**()  $\Rightarrow$  'HALLO WELT'

"Hallo Welt".**lower**()  $\Rightarrow$  'hallo welt'

- in eine Liste zerlegen

"1, 2, 3, 4".**split**(",")  $\Rightarrow$  ['1', '2', '3', '4']

- zuschneiden

" Hallo ".**strip**()  $\Rightarrow$  'Hallo'

"..Hallo..".**lstrip**(".")  $\Rightarrow$  'Hallo..'

## Ein-/Ausgabe: Dateien in Python

```
>>> input = open("in.txt")
>>> output = open("out.txt", "w")
>>> linenr = 0
>>> while True:
...     line = input.readline()
...     if not line: break
...     linenr += 1
...     output.write("{}: {}\n".format(linenr, line))
15
>>> output.close()
```

- Dateien sind mit `open(datei, mode)` erzeugte Objekte
- Nur beim Schließen (`close`) werden alle Daten geschrieben
- Mögliche Modi (Wert von `mode`):

r oder leer	lesen
w	schreiben, Datei zuvor leeren
a	schreiben, an existierende Datei anhängen

## Ein-/Ausgabe: Dateien in Python

- `datei.read()`: Lesen der gesamten Datei als Zeichenkette
- `datei.readline()`: Lesen einer Zeile als Zeichenkette
- Je nach Bedarf mittels `split`, `int` oder `float` verarbeiten
  
- `datei.write(data)`: Zeichenkette `data` zur Datei hinzufügen
- Anders als **print** kein automatisches Zeilenende
- Bei Bedarf Zeilenumbruch mit „`\n`“
- Daten, die keine Zeichenketten sind, mittels `%`-Operator oder `str` umwandeln

## Dateien als Sequenzen

```
>>> input = open("in.txt")
>>> output = open("out.txt", "w")
>>> linenr = 0
>>> for line in input:
...     linenr += 1
...     output.write(str(linenr) + ": " + line + "\n")
>>> output.close()
```

- 
- Alternative Implementation zum vorigen Beispiel
  - Dateien verhalten sich in **for** wie Listen von Zeilen
  - Einfache zeilenweise Verarbeitung
  - Aber kein Elementzugriff usw.!
  - write: alternative, umständlichere Ausgabe mittels str-Umwandlung

## Standarddateien

- Wie in der bash gibt es auch Dateien für Standard-Eingabe, -Ausgabe und Fehler-Ausgabe
- Die Dateivariablen sind:

sys.stdin	Eingabe (etwa Tastatur)
sys.stdout	Standard-Ausgabe
sys.stderr	Fehler-Ausgabe

---

```
import sys
line = sys.stdin.readline()
sys.stderr.write("don't know what to do with {}\n".format(line))
for i in range(10):
    sys.stdout.write("{} ".format(i))
sys.stdout.write("\n")
```



## Module

```
>>> import sys
>>> print("program name is \"{}\"".format(sys.argv[0]))
program name is ""
>>> from random import random
>>> print(random())
0.5673688378559075
```

- 
- Bis jetzt haben wir einen Teil der Basisfunktionalität von Python gesehen.
  - Weitere Funktionen sind in **Module** ausgelagert
  - Manche sind nicht Teil von Python und müssen erst nachinstalliert werden
  - Die Benutzung eines installierten Moduls muss per **import** angekündigt werden („Modul laden“)
  - Hilfe: `help(modul)`, alle Funktionen: `dir(modul)`

## Das sys-Modul

- Schon vorher für Eingaben benutzt
- Stellt Informationen über Python und das laufende Programm selber zur Verfügung
- `sys.argv`: Kommandozeilenparameter,  
`sys.argv[0]` ist der Programmname
- `sys.stdin`,
- `sys.stdout`,
- `sys.stderr`: Standard-Ein-/Ausgabedateien

---

```
import sys  
  
sys.stdout.write("running {}\n".format(sys.argv[0]))  
line = sys.stdin.readline()  
sys.stderr.write("some error message\n")
```

## argparse-Modul: Parameter

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("-f", "--file", dest="filename", help="write to FILE",
metavar="FILE")
parser.add_argument("number", type=int, help="the number")
args = parser.parse_args()
```

---

- Einlesen von Kommandozeilenflags
- `add_argument` spezifiziert Parameter
- kurzer + langer Name („-f“ „--file“),
- Ohne Minus positionsabhängiger Parameter
- `dest`: Zielvariable für den vom Benutzer gegebenen Wert
- `type`: geforderter Datentyp (`type="int"`)
- Bei Aufruf `python parse.py -f test 1` ist `args.filename = 'test'`, `args.number = 1`
- `python parse.py -f test a` gibt Fehler, da „a“ keine

## math- und random-Modul

```
import math
import random
def boxmuller():
    """
    calculate Gaussian random numbers using the Box-Muller transform
    """
    r1, r2 = random.random(), random.random()
    return math.sqrt(-2*math.log(r1))*math.cos(2*math.pi*r2)
```

- 
- math stellt viele mathematische Grundfunktionen zur Verfügung, z.B. **floor/ceil**, **exp/log**, **sin/cos**, **pi**
  - random erzeugt pseudozufällige Zahlen
    - **random()**: gleichverteilt in [0;1)
    - **randint(a, b)**: gleichverteilt ganze Zahlen in [a; b]
    - **gauss(m, s)**: normalverteilt mit Mittelwert m und Varianz s

## os-Modul: Betriebssystemfunktionen

```
import os
import os.path
# Datei in Verzeichnis "alt" verschieben
dir = os.path.dirname(file)
name = os.path.basename(file)
altdir = os.path.join(dir, "alt")
# Verzeichnis "alt" erstellen, falls es nicht existiert
if not os.path.isdir(altdir): os.mkdir(altdir)
# Verschieben, falls nicht schon existent
newpath = os.path.join(altdir, name)
if not os.path.exists(newpath): os.rename(file, newpath)
```

---

- Betriebssystemunabhängige Pfadtools im Untermodul os.path:  
z.B. dirname, basename, join, exists, isdir
- os.system: Programme wie von der Shell aufrufen
- os.rename/os.remove: Dateien umbenennen / löschen
- os.mkdir/os.rmdir: erzeugen / entfernen von Verzeichnissen

## subprocess-Modul: Prozesse ausführen

```
import subprocess
subprocess.run(["ls", "-l"])
subprocess.run(["ls -l", shell=True])
subprocess.run(["ls", "-l", "/dev/null"], capture_output=True)
CompletedProcess(args=['ls', '-l', '/dev/null'], returncode=0,
stdout=b'crw-rw-rw- 1 root root 1, 3 Jan 23 16:23 /dev/null\n', stderr=b'')
output = subprocess.check_output("cat list.txt| awk '{print $2}'", shell=True)
print(output)
```

---

- Mit subprocess ist es möglich neue Prozesse per spawn zu kreieren und sich mit deren input/output/error-Kanälen zu verbinden um die return-Codes zu erhalten.
- **subprocess.check\_output** gibt den StdOut zurück
- **shell=True** erlaubt einem den Befehl als einen String zu übergaben und nicht jeden Parameter durch Komma getrennt.

## GPIO-Modul: PinIO ansprechen

```
#Python Raspberry Pi GPIO Klasse importieren
import RPi.GPIO as GPIO

# Festlegung der Nutzung der vorgegebenen Nummerierung der GPIOs
GPIO.setmode(GPIO.BCM)

# GPIO17 (Pin 11) als Ausgang setzen
GPIO.setup(17, GPIO.OUT)

# GPIO17 (Pin 11) einschalten
GPIO.output(17, True)

# GPIO17 (Pin 11) ausschalten
GPIO.output(17, false)

# GPIO18 (Pin 12) als Eingang setzen
GPIO.setup(18, GPIO.IN)

# GPIO18 (Pin 12) lesen und ausgeben
value18 = GPIO.input(18)

print(value18)
```

## Bitweise Operatoren

&	AND	1, wenn beide Bits 1 sind
	OR	1, wenn eines von zwei Bits 1 ist
^	XOR	1, wenn nur eines von zwei Bits 1 ist
~	NOT	Negation, Invertiert alle Bits
<<	Links	Bitweise Verschiebung nach links
>>	Rechts	Bitweise Verschieben nach rechts

- Um Binär nur die letzte Stelle angezeigt zu bekommen einfach &1
- $x \ll y$  Ist das selbe wie  $x * 2^{**}y$
- $x \gg y$  Ist das selbe wie  $x / 2^{**}y$

$0b10 \ll 2 \Rightarrow 0b1000$

$0b101 \gg 2 \Rightarrow 0b1$



## spi-Modul

- import spidev
- spi = spidev.SpiDev()
- spi.open(0,0)

Methoden	Funktion
spi.open(0,0)	Öffnet den SPI-Bus 0 mit CS0
spi.open(0,1)	Öffnet den SPI-Bus 0 mit CS1
spi.close()	Schließt den SPI-Bus
spi.readbytes(len)	Liest len Bytes vom SPI-Slave
spi.writebytes([array of bytes])	Sendet ein Byte-Array zum SPI-Slave
spi.xfer([array of bytes], frequenz)	Sendet ein Byte-Array, CEx wird vor jedem Byte aktiv und dann wieder inaktiv

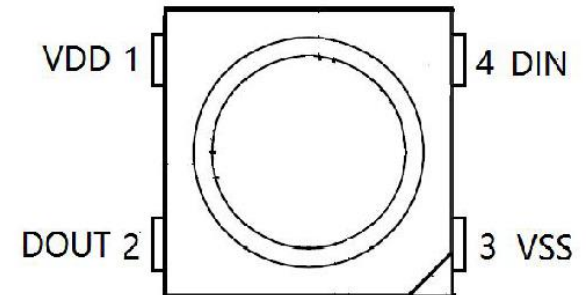
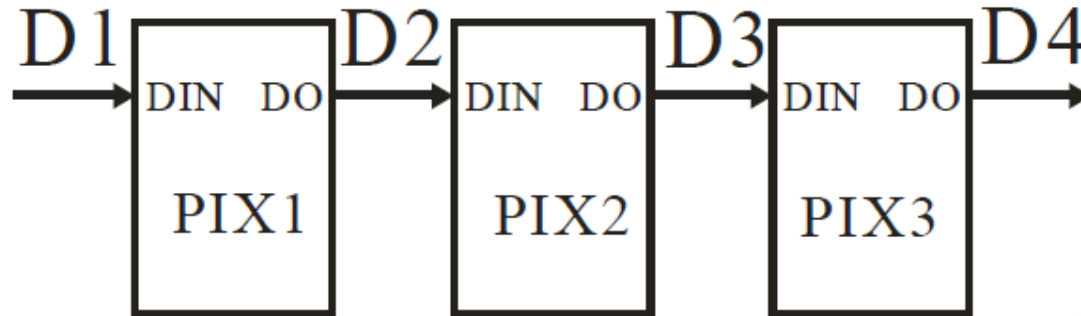
Beachte, die Schnittstelle muss über raspi-config zuvor aktiviert werden.

## WS2812b

- RGB-LED 5050
- Kontroll Chip integriert
- Für jede Farbe 1 Byte =  $2^8 = 256$  Helligkeitsstufen
- 2 kHz = 500µs
- 800 kbps

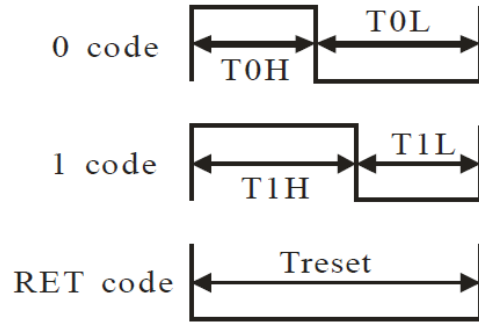
## WS2812b

- Reihenschaltung



# WS2812b

## Sequenzdiagramm



Dauer für ein Bit 1,25  $\mu$ s

8 Bit

Takt ist  $8 / 1,25 \mu\text{s} = 6,4 \text{ MHz}$

Zum Senden einer 0:

0b10000000

0x80

Zum Senden einer 1:

0b11111000

0xF8

## Data Transfer Time

T0H	0 code, high voltage time	220ns~380ns
T1H	1 code, high voltage time	580ns~1 $\mu$ s
T0L	0 code, low voltage time	580ns~1 $\mu$ s
T1L	1 code, low voltage time	220ns~420ns
RES	Frame unit, low voltage time	>280 $\mu$ s

## Aufgabe

- Bringt die RGB LEDs zum Leuchten

## Lösung

```
#!/usr/bin/python3
def write2812(spi, color):
    tx=[]
    for rgb in color:
        for byte in [rgb[1],rgb[0],rgb[2]]:
            for ibit in range(7,-1,-1):
                tx.append( 0x80 if (byte>>ibit)&1 == 0 else 0xF8 )
    spi.xfer(tx, int(8/1.25e-6))

import spidev
spi = spidev.SpiDev()
spi.open(0,0)
nLED=8
write2812(spi, [[0,5,25]]*nLED)
```

## Lösung 2

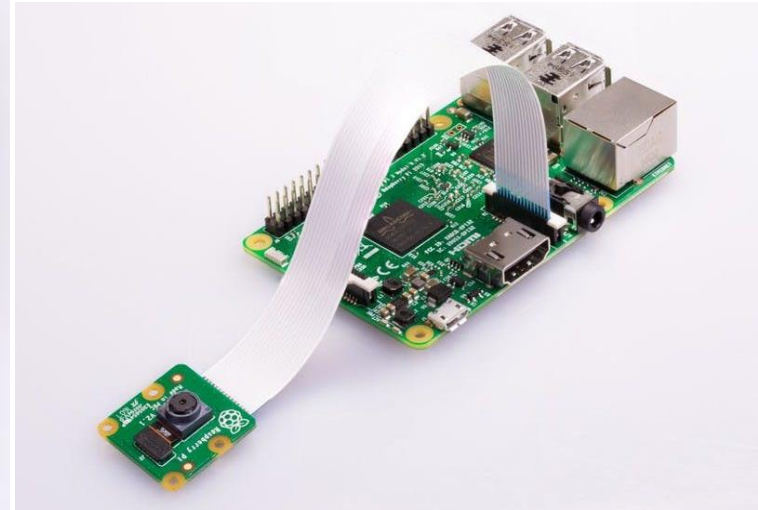
```
#!/usr/bin/python3

def write2812(spi, color):
    tx=[]
    for rgb in color:
        print(rgb)
        for byte in [rgb[1],rgb[0],rgb[2]]:
            print(byte)
            for ibit in range(7,-1,-1):
                if (byte>>ibit)&1 == 0:
                    tx.append(0X80)
                else:
                    tx.append(0xF8)
            print(tx)
    spi.xfer(tx, int(8/1.25e-6))

import spidev
spi = spidev.SpiDev()
spi.open(0,0)
nLED=8
write2812(spi, [[0,5,25]]*nLED)
```

## Raspberry Pi Camera Module V2

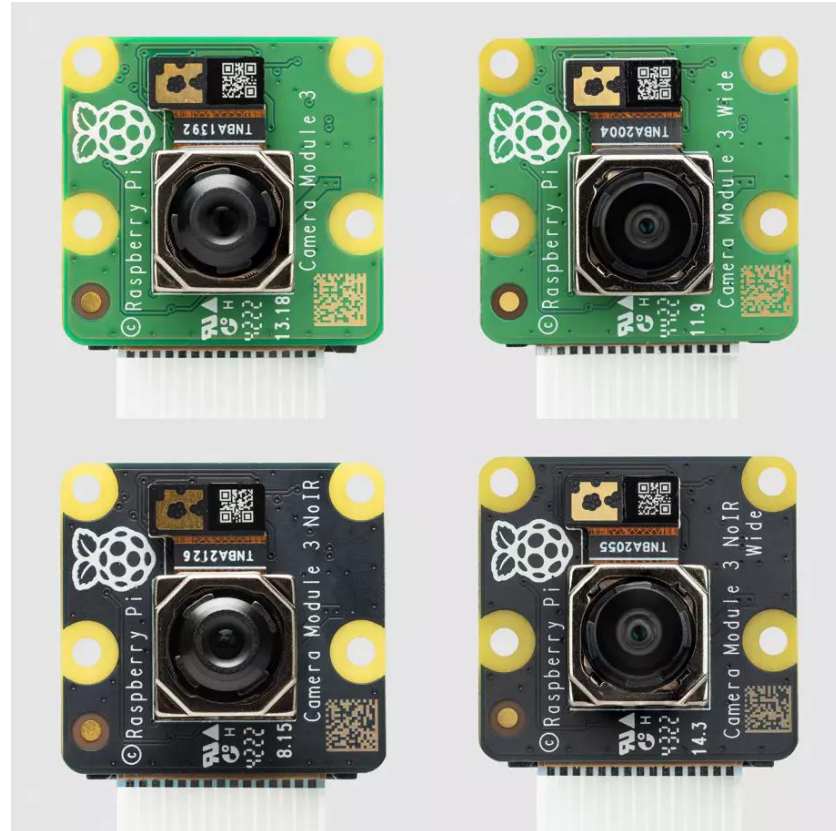
- Erschien April 2016
- Sony IMX219 8-megapixel sensor
- Foto: 3280 x 2464 Pixel
- Video: 1080p @ 30 fps
- Video: 720p @ 60 fps
- Preis: ca. 25,- €





# Raspberry Pi Camera Module V3

- Erschien Januar 2023
- Sony IMX708 12-megapixel sensor
- Autofocus
- Foto: 4608 × 2592 Pixel
- Video: 1080p @ 50 fps
- Video: 720p @ 100 fps
- Video: 480p @ 120 fps
- Blickwinkel 75° / 120°
- Preis: ca. 25,- / 35,- €



## Raspberry Pi High Quality Camera

- Erschien 2020
- Sony IMX477R 12,3-megapixel sensor
- Foto: 4056 × 3040 Pixel
- Video: 4k @ 60 fps
- Video: 1080p @ 240 fps
- Preis: ~ 55,- € ohne Objektiv



## Raspi Cam

- raspi-config Kamera aktivieren
- Bilder abspeichern mit  
`raspistill -o bild1.jpg`
- Videos abspeichern mit  
`raspivid -t 100000 -o video1.h264`  
wobei t die Aufnahmedauer in ms ist

## Raspi Cam als Überwachungskamera

- Software installieren

```
sudo apt install motion
```

- Kernel Module händisch laden

```
sudo modprobe v4l2_common  
sudo modprobe bcm2835-v4l2
```

- Kernel Module beim nächsten Boot automatisch laden

Editieren von /etc/modules und folgende Zeilen unten anhängen

```
sudo nano /etc/modules  
v4l2_common  
bcm2835-v4l2
```

- Motion beim nächsten Start automatisch Starten

```
sudo nano /etc/default/motion
```

Hier **start\_motion\_daemon** auf **Yes** setzen.

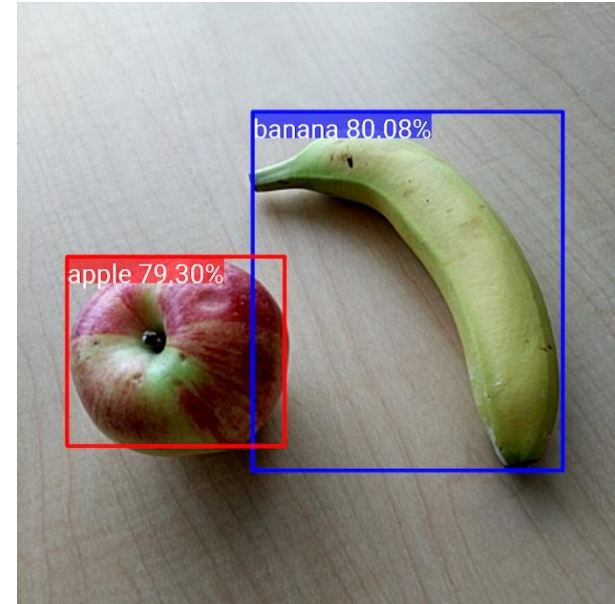
## Motion konfigurieren

■ `sudo nano /etc/motion/motion.conf`

<code>daemon on</code>	<code>#Dass Motion als Service im Hintergrund läuft</code>
<code>target_dir /home/pi/cam</code>	<code>#Verzeichnis in dem die Bilder und Videos abgelegt werden</code>
<code>stream_localhost off</code>	<code>#So dass man über <a href="http://[raspiip]:8081">http://[raspiip]:8081</a> den Live View sieht</code>
<code>width 1280</code>	<code>#Breite des Videos (Pixel)</code>
<code>height 960</code>	<code>#Höhe des Videos (Pixel)</code>
<code>framerate 2</code>	<code>#Empfangene Bilderrate pro Sekunde</code>
<code>threshold 5000</code>	<code>#Empfindlichkeit der Bewegungserkennung (Anzahl Pixel)</code>

# Tensorflow Lite

<https://www.tensorflow.org/>



<https://teachablemachine.withgoogle.com/train/image>

## Alex Sprachsteuerung auf Raspberry

- Zuerst braucht man einen Entwickleraccount bei Amazon
- <https://developer.amazon.com>
- Auf dem Raspberry flask-ask
- `sudo pip3 install flask-ask`

# Beispiel Alexa mit Flask

```
#!/usr/bin/python3
import logging
import os

from flask import Flask
from flask_ask import Ask, request, session, question, statement
#import RPi.GPIO as GPIO

app = Flask(__name__)
ask = Ask(app, "/")
logging.getLogger('flask_ask').setLevel(logging.DEBUG)

@ask.launch
def launch():
    speech_text = 'Was willst du mich fragen Strom, kippen zu, kippen auf ?'
    return question(speech_text).reprompt(speech_text).simple_card(speech_text)

@ask.intent('strom')
def strom():
    leistung = float(r.get("leistung")) / 10
    print("Da will jemand den Stromzaehler abfragen: " + str(leistung))
    return statement('Dirk du hast momentan ' + str(leistung) + ' Watt Netzbezug')

if __name__ == '__main__':
    if 'ASK_VERIFY_REQUESTS' in os.environ:
        verify = str(os.environ.get('ASK_VERIFY_REQUESTS', '')).lower()
        if verify == 'false':
            app.config['ASK_VERIFY_REQUESTS'] = False
    app.run(debug=True,host='0.0.0.0',port=8011)
```



## InfluxDB

InfluxDB ist eine Time Series Database (TSDB)

- speziell für Zeitreihen
- SQL ähnlich

Vorteil gegenüber andere Datenbanken:

- Bessere Performance und geringerer Ressourcenbedarf
- Spezielle Funktionen für das Downsampling von Messreihen
- Bessere Kompression von Daten

## InfluxDB Installation

```
sudo apt install influxdb  
sudo apt install python3-influxdb  
sudo apt install influxdb-client
```

- Setzen des admin Passwortes

```
influx  
CREATE USER admin WITH PASSWORD 'geheim' WITH ALL PRIVILEGES
```

- Anzeigen der eingerichteten Benutzer

```
SHOW USERS
```

- Datenbank anlegen

```
CREATE DATABASE raspi
```

## Python3 InfluxDB

```
#InfluxDB
from influxdb import InfluxDBClient

host='127.0.0.1'
port='8086'
user='admin'
password='geheim'
dbname = 'raspi'
clientinflux = InfluxDBClient(host, port, user, password, dbname)
...
# Send the JSON data to InfluxDB
fehler = clientinflux.write_points(data,time_precision='s')
print(fehler)
```

# JSON Format

## JavaScript Object Notation

- kompaktes Datenformat in einer einfach lesbaren Textform

## Datentypen:

Nullwert            wird durch das Schlüsselwort null dargestellt.

Boolescher Wert   wird durch die Schlüsselwörter true und false dargestellt.

Zahl                ist eine Folge der Ziffern 0–9. [ - . e E ]  
Beispiele: 2 , -78568 , -123.75 , 2e+6 , 2E-3

Zeichenkette      beginnt und endet mit doppelten geraden Anführungszeichen (").  
Sie kann Unicode-Zeichen und durch \ eingeleitete Escape-Sequenzen enthalten.

Array                beginnt mit [ und endet mit ]. Es enthält eine durch Kommata geteilte, geordnete Liste von Elementen gleichen oder verschiedenen Typs. Leere Arrays sind zulässig.

Objekt              beginnt mit { und endet mit }. Es enthält eine durch Kommata geteilte, ungeordnete Liste von Eigenschaften. Objekte ohne Eigenschaften („leere Objekte“) sind zulässig.

### Eigenschaft

besteht aus einem Schlüssel und einem Wert, getrennt durch einen Doppelpunkt (Schlüssel : Wert). Die Schlüssel sollten eindeutig sein, da unterschiedliche Parser mit mehrfach vorkommenden Schlüsseln unterschiedlich umgehen.

der **Schlüssel** ist eine Zeichenkette.

der **Wert** ist ein beliebiges Element.

# JSON Beispiel

```
[
  {
    "firstName": "Jane",
    "lastName": "Doe",
    "hobbies": ["running", "sky diving", "singing"],
    "age": 35,
    "children": [
      {
        "firstName": "Alice",
        "age": 6
      },
      {
        "firstName": "Bob",
        "age": 8
      }
    ]
  }
]
```

## InfluxDB JSON

- Es werden drei Schlüssel benötigt
  - measurement
  - time
  - fields

Die aktuelle Zeit mit Python bekommen:

```
time.asctime(time.gmtime())
```

## Aufgabe

- Ändere das Skript das den Bosch Sensor ausliest so weit, dass er zusätzlich auch die Daten in die InfluxDB schreibt.

## Lösung

- Das fehlende JSON Teil:

```
# Create the JSON data structure
measurement = 'umwelt'
iso = time.asctime(time.gmtime())
data = [
    {
        "measurement": measurement,
        "time": iso,
        "fields": {
            "Temperatur" : temperature,
            "Luftdruck"   : pressure
        }
    }
]
```



## Grafana

- `wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -`
- `echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list`
- `sudo apt update`
- `sudo apt install grafana`
- `sudo systemctl enable grafana-server`
- `sudo systemctl start grafana-server`
- `http://<ip address>:3000 admin admin`